Circuit Simulation

8

8.1 Introduction

Fabricating chips is expensive and time-consuming, so designers need simulation tools to explore the design space and verify designs before they are fabricated. Simulators operate at many levels of abstraction, from process through architecture. *Process simulators* such as SUPREME predict how factors in the process recipe such as time and temperature affect device physical and electrical characteristics. *Circuit simulators* such as SPICE and Spectre use device models and a circuit netlist to predict circuit voltages and currents, which indicate performance and power consumption. *Logic simulators* such as VCS and ModelSim are widely used to verify correct logical operation of designs specified in a hardware description language (HDL). *Architecture simulators*, sometimes offered with a processor's development toolkit, work at the level of instructions and registers to predict throughput and memory access patterns, which influence design decisions such as pipelining and cache memory organization. The various levels of abstraction offer trade-offs between degree of detail and the size of the system that can be simulated. VLSI designers are primarily concerned with circuit and logic simulation. This chapter focuses on circuit simulation with SPICE. Section 15.3 discusses logic simulation.

Is it better to predict circuit behavior using paper-and-pencil analysis, as has been done in the previous chapters, or with simulation? VLSI circuits are complex and modern transistors have nonlinear, nonideal behavior, so simulation is necessary to accurately predict detailed circuit behavior. Even when closed-form solutions exist for delay or transfer characteristics, they are too time-consuming to apply by hand to large numbers of circuits. On the other hand, circuit simulation is notoriously prone to errors: *garbage in, garbage out* (GIGO). The simulator accepts the model of reality provided by the designer, but it is very easy to create a model that is inaccurate or incomplete. Moreover, the simulator only applies the stimulus provided by the designer, and it is common to overlook the worst-case stimulus. In the same way that an experienced programmer doesn't expect a program to operate correctly before debugging, an experienced VLSI designer does not expect that the first run of a simulation will reflect reality. Therefore, the circuit designer needs to have a good intuitive understanding of circuit operation and should be able to predict the expected outcome before simulating. Only when expectation and simulation match can there be confidence in the results. In practice, circuit designers depend on both hand analysis and simulation, or as [Glasser85] puts it, "simulation guided through insight gained from analysis."

This chapter presents a brief SPICE tutorial by example. It then discusses models for transistors and diffusion capacitance. The remainder of the chapter is devoted to simulation techniques to characterize a process and to check performance, power, and correctness of circuits and interconnect.

8.2 A SPICE Tutorial

SPICE (*Simulation Program with Integrated Circuit Emphasis*) was originally developed in the 1970s at Berkeley [Nagel75]. It solves the nonlinear differential equations describing components such as transistors, resistors, capacitors, and voltage sources. SPICE offers many ways to analyze circuits, but digital VLSI designers are primarily interested in *DC* and *transient* analysis that predicts the node voltages given inputs that are fixed or arbitrarily changing in time. SPICE was originally developed in FORTRAN and has some idiosyncrasies, particularly in file formats, related to its heritage. There are free versions of SPICE available on most platforms, but the commercial versions tend to offer more robust numerical convergence. In particular, HSPICE is widely used in industry because it converges well, supports the latest device and interconnect models, and has a large number of enhancements for measuring and optimizing circuits. PSPICE is another commercial version with a free limited student version. LTSpice is a robust free version. The examples throughout this section use HSPICE and generally will not run in ordinary SPICE.

While the details of using SPICE vary with version and platform, all versions of SPICE read an input file and generate a list file with results, warnings, and error messages. The input file is often called a *SPICE deck* and each line a *card* because it was once provided to a mainframe as a deck of punch cards. The input file contains a netlist consisting of components and nodes. It also contains simulation options, analysis commands, and device models. The netlist can be entered by hand or extracted from a circuit schematic or layout in a CAD program.

A good SPICE deck is like a good piece of software. It should be readable, maintainable, and reusable. Comments and white space help make the deck readable. Often, the best way to write a SPICE deck is to start with a good deck that does nearly the right thing and then modify it.

The remainder of this section provides a sequence of examples illustrating the key syntax and capabilities of SPICE for digital VLSI circuits. For more detail, consult the Berkeley SPICE manual [Johnson91], the lengthy HSPICE manual, or any number of textbooks on SPICE (such as [Kielkowski95, Foty96]).

8.2.1 Sources and Passive Components

Suppose we would like to find the response of the RC circuit in Figure 8.1(a) given an input rising from 0 to 1.0 V over 50 ps. Because the RC time constant of 100 fF $\times$ 2 kΩ = 200 ps is much greater than the input rise time, we intuitively expect the output would look like an exponential asymptotically approaching the final value of 1.0 V with a 200 ps time constant. Figure 8.2 gives a SPICE deck for this simulation and Figure 8.1(b) shows the input and output responses.

Lines beginning with * are comments. The first line of a SPICE deck must be a comment, typically indicating the title of the simulation. It is good practice to treat SPICE input files like computer programs and follow similar procedures for commenting the decks. In particular, giving the author, date, and objective of the simulation at the beginning is helpful when the deck must be revisited in the future (e.g., when a chip is in silicon

debug and old simulations are being reviewed to track down potential reasons for failure).

Control statements begin with a dot (`.`). The `.option post` statement instructs HSPICE to write the results to a file for use with a waveform viewer. The last statement of a SPICE deck must be `.end`.

Each line in the netlist begins with a letter indicating the type of circuit element. Common elements are given in Table 8.1. In this case, the circuit consists of a voltage source named `Vin`, a resistor named `R1`, and a capacitor named `C1`. The nodes in the circuit are named `in`, `out`, and `gnd`. `gnd` is a special node name defined to be the 0 V reference. The units consist of one or two letters. The first character indicates the order of magnitude, as given in Table 8.2. Take note that mega is `x`, not `m`. The second letter indicates a unit for human convenience (such as `F` for farad or `s` for second) and is ignored by SPICE. For example, the hundred femtofarad capacitor can be expressed as `100fF`, `100f`, or simply `100e-15`. Note that SPICE is case-insensitive but consistent capitalization is good practice nonetheless because the netlist might be parsed by some other tool.

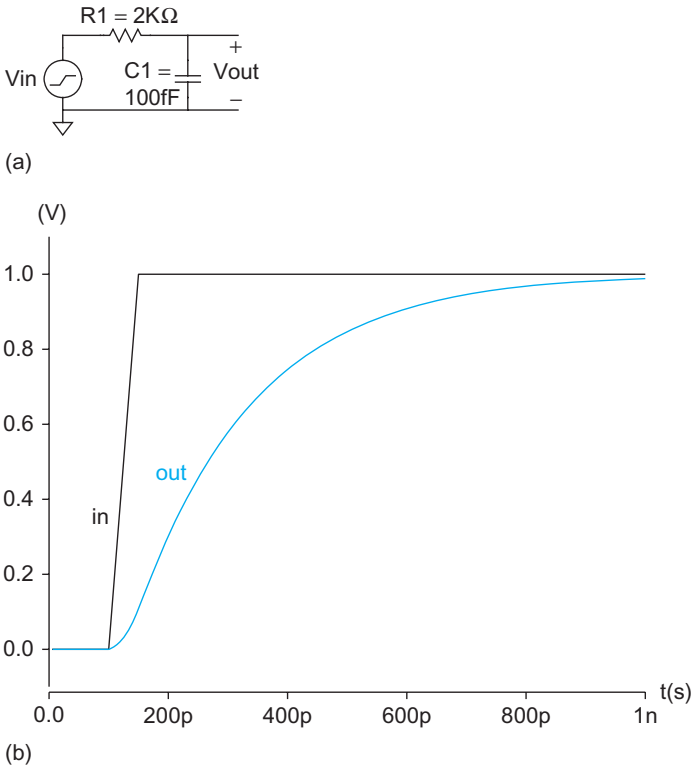

FIGURE 8.1 RC circuit response

```
* rc.sp
* David_Harris@hmc.edu 2/2/03
* Find the response of RC circuit to rising input

*---------------------------------------------------------------
* Parameters and models
*---------------------------------------------------------------
.option post

*---------------------------------------------------------------
* Simulation netlist
*---------------------------------------------------------------
Vin    in     gnd    pwl    0ps 0 100ps 0 150ps 1.0 1ns 1.0
R1     in     out    2k
C1     out    gnd    100f

*---------------------------------------------------------------
* Stimulus
*---------------------------------------------------------------
.tran 20ps 1ns
.plot v(in) v(out)
.end
```

FIGURE 8.2 RC SPICE deck

TABLE 8.1 Common SPICE elements

Letter	Element
R	Resistor
C	Capacitor
L	Inductor
K	Mutual inductor
V	Independent voltage source
I	Independent current source
M	MOSFET
D	Diode
Q	Bipolar transistor
W	Lossy transmission line
X	Subcircuit
E	Voltage-controlled voltage source
G	Voltage-controlled current source
H	Current-controlled voltage source
F	Current-controlled current source

TABLE 8.2 SPICE units

Letter	Unit	Magnitude
a	atto	10^{-18}
f	femto	10^{-15}
p	pico	10^{-12}
n	nano	10^{-9}
u	micro	10^{-6}
m	milli	10^{-3}
k	kilo	10^{3}
x	mega	10^{6}
g	giga	10^{9}

The voltage source is defined as a piecewise linear (PWL) source. The waveform is specified with an arbitrary number of (time, voltage) pairs. Other common sources include DC sources and pulse sources. A DC voltage source named `Vdd` that sets node `vdd` to 2.5 V could be expressed as

```
Vdd   vdd   gnd   2.5
```

Pulse sources are convenient for repetitive signals like clocks. The general form for a pulse source is illustrated in Figure 8.3. For example, a clock with a 1.0 V swing, 800 ps period, 100 ps rise and fall times, and 50% duty cycle (i.e., equal high and low times) would be expressed as

```
Vck   clk   gnd   PULSE 0 1 0ps 100ps 100ps 300ps 800ps
```

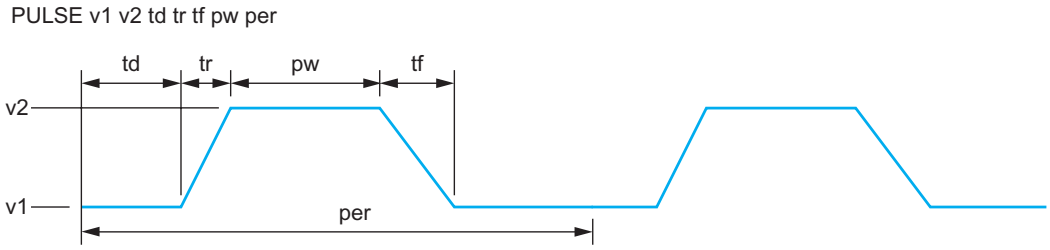

FIGURE 8.3 Pulse waveform

The stimulus specifies that a transient analysis (`.tran`) should be performed using a maximum step size of 20 ps for a duration of 1 ns. When plotting node voltages, the step size determines the spacing between points.

The `.plot` command generates a textual plot of the node variables specified (in this case the voltages at nodes `in` and `out`), as shown in Figure 8.4. Similarly, the `.print` statement prints the results in a multicolumn table. Both commands show the legacy of

```
legend:

  a: v(in)
  b: v(out)

      time        v(in)
 (ab      )     -500.0000m        0.           500.0000m        1.0000        1.5000
                    +                +                +                +             +
    0.          0.     -+------+------2------+------+------+------+------+------+-
   20.0000p     0.      +      +       2      +      +      +      +      +      +
   40.0000p     0.      +      +       2      +      +      +      +      +      +
   60.0000p     0.      +      +       2      +      +      +      +      +      +
   80.0000p     0.      +      +       2      +      +      +      +      +      +
  100.0000p     0.      +      +       2      +      +      +      +      +      +
  120.0000p  400.000m +      +      +b      +    a +      +      +      +      +
  140.0000p  800.000m +      +      + b     +      +      +      +a     +      +
  160.0000p    1.000  +      +      +    b  +      +      +      a      +      +
  180.0000p    1.000  +      +      +       b      +      +      a      +      +
  200.0000p    1.000 -+------+------+-------+-b----+------+------+------a------+------+-
  220.0000p    1.000  +      +      +       +  b   +      +      +      a      +      +
  240.0000p    1.000  +      +      +       +    b +      +      +      a      +      +
  260.0000p    1.000  +      +      +       +      b      +      +      a      +      +
  280.0000p    1.000  +      +      +       +      +b     +      +      a      +      +
  300.0000p    1.000  +      +      +       +      + b    +      +      a      +      +
  320.0000p    1.000  +      +      +       +      +  b   +      +      a      +      +
  340.0000p    1.000  +      +      +       +      +    b +      +      a      +      +
  360.0000p    1.000  +      +      +       +      +     b+      +      a      +      +
  380.0000p    1.000  +      +      +       +      +      b      +      a      +      +
  400.0000p    1.000 -+------+------+------+------+------+------b------a------+------+-
  420.0000p    1.000  +      +      +       +      +      +b     +      a      +      +
  440.0000p    1.000  +      +      +       +      +      +b     +      a      +      +
  460.0000p    1.000  +      +      +       +      +      + b    +      a      +      +
  480.0000p    1.000  +      +      +       +      +      + b    +      a      +      +
  500.0000p    1.000  +      +      +       +      +      +  b   +      a      +      +
  520.0000p    1.000  +      +      +       +      +      +  b   +      a      +      +
  540.0000p    1.000  +      +      +       +      +      +   b  +      a      +      +
  560.0000p    1.000  +      +      +       +      +      +   b  +      a      +      +
  580.0000p    1.000  +      +      +       +      +      +   b  +      a      +      +
  600.0000p    1.000 -+------+------+------+------+------+------+---b--a------+------+-
  620.0000p    1.000  +      +      +       +      +      +      b a     +      +
  640.0000p    1.000  +      +      +       +      +      +      b a     +      +
  660.0000p    1.000  +      +      +       +      +      +      b a     +      +
  680.0000p    1.000  +      +      +       +      +      +      b a     +      +
  700.0000p    1.000  +      +      +       +      +      +      b a     +      +
  720.0000p    1.000  +      +      +       +      +      +       ba     +      +
  740.0000p    1.000  +      +      +       +      +      +       ba     +      +
  760.0000p    1.000  +      +      +       +      +      +       ba     +      +
  780.0000p    1.000  +      +      +       +      +      +       ba     +      +
  800.0000p    1.000 -+------+------+------+------+------+------+------ba------+------+-
  820.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  840.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  860.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  880.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  900.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  920.0000p    1.000  +      +      +       +      +      +      +       ba     +      +
  940.0000p    1.000  +      +      +       +      +      +      +        2     +      +
  960.0000p    1.000  +      +      +       +      +      +      +        2     +      +
  980.0000p    1.000  +      +      +       +      +      +      +        2     +      +
    1.0000n    1.000 -+------+------+------+------+------+------+------2------+------+-
                    +                +                +                +             +
```

FIGURE 8.4 Textual plot of RC circuit response

FORTRAN and line printers. On modern computers with graphical user interfaces, the `.option post` command is usually preferred. It generates a file (in this case, `rc.tr0`)containing the results of the specified (transient) analysis. Then, a separate graphical waveform viewer can be used to look at and manipulate the waveforms. SPICE Explorer is a waveform viewer from Synopsys compatible with HSPICE.

8.2.2 Transistor DC Analysis

One of the first steps in becoming familiar with a new CMOS process is to look at the I-V characteristics of the transistors. Figure 8.5(a) shows test circuits for a unit (4/2 λ) nMOS transistor in a 65 nm process at V_{DD} = 1.0 V. The I-V characteristics are plotted in Figure 8.5(b) using the SPICE deck in Figure 8.6.

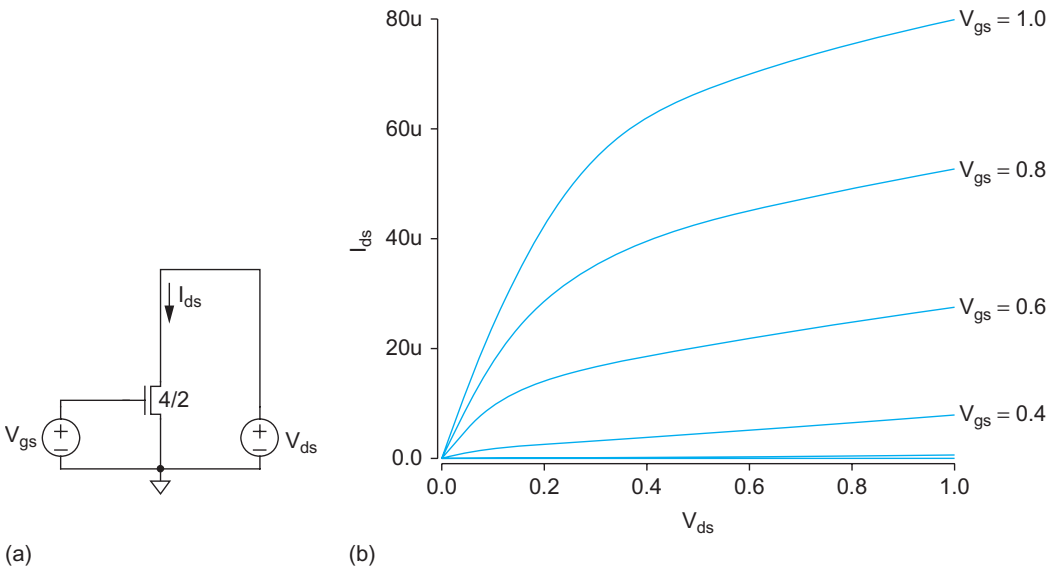

FIGURE 8.5 MOS I-V characteristics. Current in units of microamps (u).

`.include` reads another SPICE file from disk. In this example, it loads device models that will be discussed further in Section 8.3. The circuit uses two independent voltage sources with default values of 0 V; these voltages will be varied by the `.dc` command. The nMOS transistor is defined with the MOSFET element `M` using the syntax

```
Mname   drain   gate   source   body   model   W=<width>   L=<length>
```

Note that this process has λ = 25 nm and a minimum drawn channel length of 50 nm even though it is nominally called a 65 nm process.

The `.dc` command varies the voltage source `Vgs` DC voltage from 0 to 1.0 V in increments of 0.05 V. This is repeated multiple times as `Vgs` is swept from 0 to 1.0 V in 0.2 V increments to compute many I_{ds} vs. V_{ds} curves at different values of V_{gs}.

8.2.3 Inverter Transient Analysis

Figure 8.7 shows the step response of an unloaded unit inverter, annotated with propagation delay and 20–80% rise and fall times. Observe that significant initial overshoot from bootstrapping

```
* mosiv.sp

*-------------------------------------------------------------------------
* Parameters and models
*-------------------------------------------------------------------------
.include '../models/ibm065/models.sp'
.temp 70
.option post

*-------------------------------------------------------------------------
* Simulation netlist
*-------------------------------------------------------------------------
*nmos
Vgs    g    gnd    0
Vds    d    gnd    0
M1     d    g      gnd    gnd    NMOS    W=100n   L=50n

*-------------------------------------------------------------------------
* Stimulus
*-------------------------------------------------------------------------
.dc Vds 0 1.0 0.05 SWEEP Vgs 0 1.0 0.2
.end
```

FIGURE 8.6 MOSIV SPICE deck

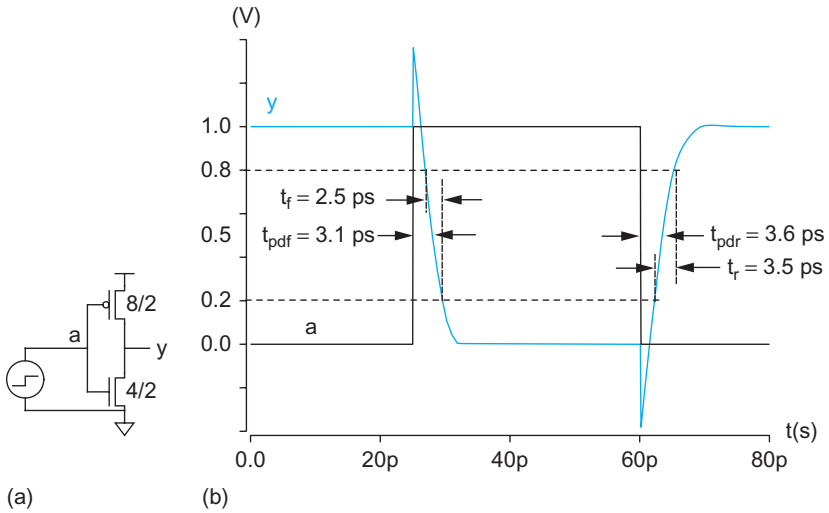

FIGURE 8.7 Unloaded inverter

occurs because there is no load (see Section 4.4.6.6). The SPICE deck for the simulation is shown in Figure 8.8.

This deck introduces the use of parameters and scaling. The `.param` statement defines a parameter named `SUPPLY` to have a value of 1.0. This is then used to set `Vdd` and the amplitude of the input pulse. If we wanted to evaluate the response at a different supply voltage, we would simply need to change the `.param` statement. The `.scale` sets a scale factor for all dimensions that would by default be measured in meters. In this case, it sets the scale to $\lambda = 25$ nm. Now the transistor widths and lengths in the inverter are specified in terms of

```
* inv.sp

*-----------------------------------------------------------------------
* Parameters and models
*-----------------------------------------------------------------------
.param SUPPLY=1.0
.option scale=25n
.include '../models/ibm065/models.sp'
.temp 70
.option post

*-----------------------------------------------------------------------
* Simulation netlist
*-----------------------------------------------------------------------
Vdd   vdd   gnd   'SUPPLY'
Vin   a     gnd   PULSE  0 'SUPPLY' 25ps 0ps 0ps 35ps 80ps
M1    y     a     gnd    gnd   NMOS   W=4   L=2
+ AS=20 PS=18 AD=20 PD=18
M2    y     a     vdd    vdd   PMOS   W=8   L=2
+ AS=40 PS=26 AD=40 PD=26

*-----------------------------------------------------------------------
* Stimulus
*-----------------------------------------------------------------------
.tran 0.1ps 80ps
.end
```

FIGURE 8.8 INV SPICE deck

lambda rather than in meters. This is convenient for chips designed using scalable rules, but is not normally done in commercial processes with micron-based rules.

Recall that parasitic delay is strongly dependent on diffusion capacitance, which in turn depends on the area and perimeter of the source and drain. As each diffusion region in an inverter must be contacted, the geometry resembles that of Figure 2.8(a). The diffusion width equals the transistor width and the diffusion length is 5 λ. Thus, the area of the source and drain are AS = AD = $5W\lambda^2$ and the perimeters are PS = PD = $(2W + 10)\lambda$. Note that the + sign in the first column of a line indicates that it is a continuation of the previous line. These dimensions are also affected by the scale factor.

8.2.4 Subcircuits and Measurement

One of the simplest measures of a process's inherent speed is the fanout-of-4 inverter delay. Figure 8.9(a) shows a circuit to measure this delay. The nMOS and pMOS transistor sizes (in multiples of a unit 4/2 λ transistor) are listed below and above each gate, respectively. $X3$ is the inverter under test and $X4$ is its load, which is four times larger than $X3$. To first order, these two inverters would be sufficient. However, the delay of $X3$ also depends on the input slope, as discussed in Section 4.4.6.1. One way to obtain a realistic input slope is to drive node `c` with a pair of FO4 inverters $X1$ and $X2$. Also, as discussed in Section 4.4.6.6, the input capacitance of $X4$ depends not just on its C_{gs} but also on C_{gd}. C_{gd} is Miller-multiplied as node `e` switches and would be effectively doubled if `e` switched instantaneously. When `e` is loaded with $X5$, it switches at a slower, more realistic rate, slightly reducing the effective capacitance presented at node `d` by $X4$. The waveforms in Figure 8.9(b) are annotated with the rising and falling delays.

SPICE decks are easier to read and maintain when common circuit elements are captured as subcircuits. For example, the deck in Figure 8.10 computes the FO4 inverter delay using an inverter subcircuit.

The `.global` statement defines `vdd` and `gnd` as global nodes that can be referenced from within subcircuits. The inverter is declared as a subcircuit with two terminals: `a` and `y`. It also accepts two parameters specifying the width of the nMOS and pMOS transistors; these parameters have default values of 4 and 8, respectively. The source and drain area and perimeter are functions of the transistor widths. HSPICE evaluates functions given inside single quotation marks. The functions can include parameters, constants, parentheses, +, –, *, /, and ** (raised to a power).

The simulation netlist contains the power supply, input source, and five inverters. Each inverter is a subcircuit (`X`) element. As `N` and `P` are not specified, each uses the default size. The `M` parameter multiplies all the currents in the subcircuit by the factor given, equivalent to `M` elements wired in parallel. In this case, the fanouts are expressed in terms of a parameter `H`. Thus, *X*2 has the capacitance and output current of 4 unit inverters, while *X*3 is equivalent to 16. Another way to model the inverters would have been to use the `N` and `P` parameters:

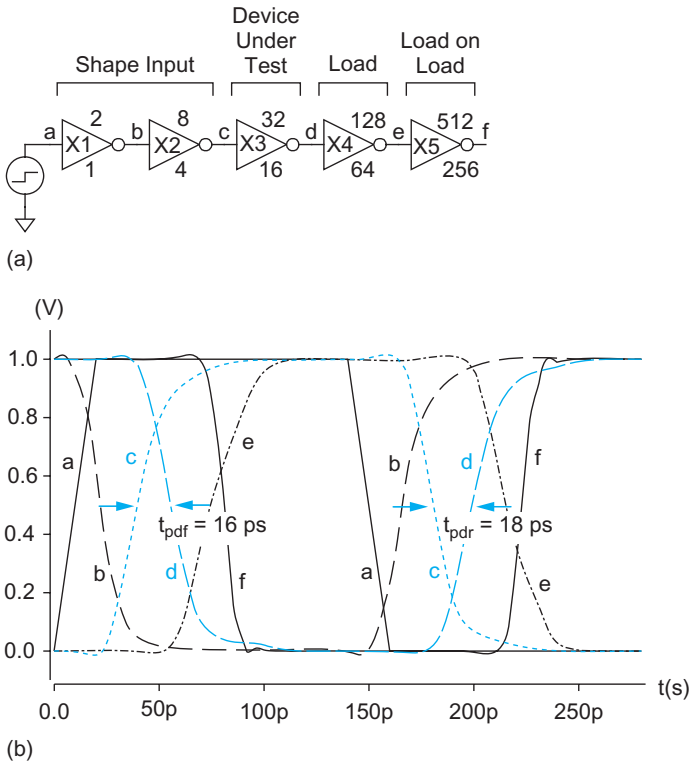

FIGURE 8.9 Fanout-of-4 inverters

```
X1    a    b    inv    N=4      P=8      * shape input waveform
X2    b    c    inv    N=16     P=32     * reshape input waveform
X3    c    d    inv    N=64     P=128    * device under test
X4    d    e    inv    N=256    P=512    * load
X5    e    f    inv    N=1024   P=2048   * load on load
```

However, a transistor of four times unit width does not have exactly the same input capacitance or output current as four unit inverters tied in parallel, so the `M` parameter is preferred.

In this example, the subcircuit declaration and simulation netlist are part of the SPICE deck. When working with a standard cell library, it is common to keep subcircuit declarations in their own files and reference them with a `.include` statement instead. When the simulation netlist is extracted from a schematic or layout CAD system, it is common to put the netlist in a separate file and `.include` it as well.

The `.measure` statement measures simulation results and prints them in the listing file. The deck measures the rising propagation delay t_{pdr} as the difference between the time that the input `c` first falls through $V_{DD}/2$ and the time that the output `d` first rises through $V_{DD}/2$. `TRIG` and `TARG` indicate the trigger and target events between which delay is measured. The `.measure` statement can also be used to compute functions of other measurements. For example, the average FO4 inverter propagation delay t_{pd} is the mean of t_{pdr} and t_{pdf}, 17 ps. The 20–80% rise time is $t_r = 20$ ps and the fall time is $t_f = 17$ ps.

```
* fo4.sp
*-----------------------------------------------------------------------
* Parameters and models
*-----------------------------------------------------------------------
.param SUPPLY=1.0
.param H=4
.option scale=25n
.include '../models/ibm065/models.sp'
.temp 70
.option post
*-----------------------------------------------------------------------
* Subcircuits
*-----------------------------------------------------------------------
.global vdd gnd
.subckt inv a y N=4 P=8
M1    y     a     gnd     gnd     NMOS    W='N'   L=2
+ AS='N*5' PS='2*N+10' AD='N*5' PD='2*N+10'
M2    y     a     vdd     vdd     PMOS    W='P'   L=2
+ AS='P*5' PS='2*P+10' AD='P*5' PD='2*P+10'
.ends
*-----------------------------------------------------------------------
* Simulation netlist
*-----------------------------------------------------------------------
Vdd   vdd   gnd   'SUPPLY'
Vin   a     gnd   PULSE   0 'SUPPLY' 0ps 20ps 20ps 120ps 280ps
X1    a     b     inv               * shape input waveform
X2    b     c     inv     M='H'     * reshape input waveform
X3    c     d     inv     M='H**2'  * device under test
X4    d     e     inv     M='H**3'  * load
X5    e     f     inv     M='H**4'  * load on load
*-----------------------------------------------------------------------
* Stimulus
*-----------------------------------------------------------------------
.tran 0.1ps 280ps
.measure tpdr                                 * rising prop delay
+     TRIG v(c)    VAL='SUPPLY/2' FALL=1
+     TARG v(d)    VAL='SUPPLY/2' RISE=1
.measure tpdf                                 * falling prop delay
+     TRIG v(c)    VAL='SUPPLY/2' RISE=1
+     TARG v(d)    VAL='SUPPLY/2' FALL=1
.measure tpd param='(tpdr+tpdf)/2'            * average prop delay
.measure trise                                * rise time
+     TRIG v(d)    VAL='0.2*SUPPLY' RISE=1
+     TARG v(d)    VAL='0.8*SUPPLY' RISE=1
.measure tfall                                * fall time
+     TRIG v(d)    VAL='0.8*SUPPLY' FALL=1
+     TARG v(d)    VAL='0.2*SUPPLY' FALL=1
.end
```

FIGURE 8.10 FO4 SPICE deck

8.2.5 Optimization

In many examples, we have assumed that a *P*/*N* ratio of 2:1 gives approximately equal rise and fall delays. The FO4 inverter simulation showed that a ratio of 2:1 gives rising delays that are slower than the falling delays because the pMOS mobility is less than half that of the nMOS. You could repeatedly run simulations with different default values of `P` to find the ratio for equal delay. HSPICE has built-in optimization capabilities that will automat-

ically tweak parameters to achieve some goal and report what parameter value gave the best results. Figure 8.11 shows a modified version of the FO4 inverter simulation using the optimizer.

The subcircuits $X1$–$X5$ override their default pMOS widths to use a width of `P1` instead. In the optimization setup, the difference of t_{pdr} and t_{pdf} is measured. The goal of the optimization will be to drive this difference to 0. To do this, `P1` may be var-

```
* fo4opt.sp
*-----------------------------------------------------------------------
* Parameters and models
*-----------------------------------------------------------------------
.param SUPPLY=1.0
.option scale=25n
.include '../models/ibm065/models.sp'
.temp 70
.option post
*-----------------------------------------------------------------------
* Subcircuits
*-----------------------------------------------------------------------
.global vdd gnd
.subckt inv a y N=4 P=8
M1   y   a   gnd   gnd   NMOS   W='N'   L=2
+ AS='N*5' PS='2*N+10' AD='N*5' PD='2*N+10'
M2   y   a   vdd   vdd   PMOS   W='P'   L=2
+ AS='P*5' PS='2*P+10' AD='P*5' PD='2*P+10'
.ends
*-----------------------------------------------------------------------
* Simulation netlist
*-----------------------------------------------------------------------
Vdd   vdd   gnd   'SUPPLY'
Vin   a     gnd   PULSE    0 'SUPPLY' 0ps 20ps 20ps 120ps 280ps
X1    a     b     inv      P='P1'            * shape input waveform
X2    b     c     inv      P='P1'    M=4     * reshape input waveform
X3    c     d     inv      P='P1'    M=16    * device under test
X4    d     e     inv      P='P1'    M=64    * load
X5    e     f     inv      P='P1'    M=256   * load on load
*-----------------------------------------------------------------------
* Optimization setup
*-----------------------------------------------------------------------
.param P1=optrange(8,4,16)          * search from 4 to 16, guess 8
.model optmod opt itropt=30         * maximum of 30 iterations
.measure bestratio param='P1/4'     * compute best P/N ratio
*-----------------------------------------------------------------------
* Stimulus
*-----------------------------------------------------------------------
.tran 0.1ps 280ps SWEEP OPTIMIZE=optrange RESULTS=diff MODEL=optmod
.measure tpdr                              * rising propagation delay
+     TRIG v(c)   VAL='SUPPLY/2' FALL=1
+     TARG v(d)   VAL='SUPPLY/2' RISE=1
.measure tpdf                              * falling propagation delay
+     TRIG v(c)   VAL='SUPPLY/2' RISE=1
+     TARG v(d)   VAL='SUPPLY/2' FALL=1
.measure tpd param='(tpdr+tpdf)/2' goal=0  * average prop delay
.measure diff param='tpdr-tpdf' goal = 0   * diff between delays
.end
```

FIGURE 8.11 FO4OPT SPICE deck

ied from 4 to 16, with an initial guess of 8. The optimizer may use up to 30 iterations to find the best value of `P1`. Because the nMOS width is fixed at 4, the best *P*/*N* ratio is computed as `P1`/4. The transient analysis includes a `SWEEP` statement containing the parameter to vary, the desired result, and the number of iterations.

HSPICE determines that the *P*/*N* ratio for equal rise and fall delay is 2.87:1, giving a rising and falling delay of 17.9 ps. This is slower than what the 2:1 ratio provides and requires large, power-hungry pMOS transistors, so such a high ratio is seldom used.

A similar scenario is to find the *P*/*N* ratio that gives lowest average delay. By changing the `.tran` statement to use `RESULTS=tpd`, we find a best ratio of 1.79:1 with rising, falling, and average propagation delays of 18.8, 15.2, and 17.0 ps, respectively. Whenever you do an optimization, it is important to consider not only the optimum but also the sensitivity to deviations from this point. Further simulation finds that *P*/*N* ratios of anywhere from 1.5:1 to 2.2:1 all give an average propagation delay of better than 17.2 ps. There is no need to slavishly stick to the 1.79:1 "optimum." The best *P*/*N* ratio in practice is a compromise between using smaller pMOS devices to save area and power and using larger devices to achieve more nearly equal rise/fall times and avoid the hot electron reliability problems induced by very slow rising edges in circuits with weak pMOS transistors. *P*/*N* ratios are discussed further in Section 9.2.1.6.

8.2.6 Other HSPICE Commands

The full HSPICE manual fills over 4000 pages and includes many more capabilities than can be described here. A few of the most useful additional commands are covered in this section. Section 8.3 describes transistor models and library calls, and Section 8.6 discusses modeling interconnect with lossy transmission lines.

```
.option accurate
```

Tighten integration tolerances to obtain more accurate results. This is useful for oscillators and high-gain analog circuits or when results seem fishy.

```
.option autostop
```

Conclude simulation when all `.measure` results are obtained rather than continuing for the full duration of the `.tran` statement. This can substantially reduce simulation time.

```
.temp 0 70 125
```

Repeat the simulation three times at temperatures of 0, 70, and 125 °C. Device models may contain information about how changing temperature changes device performance.

```
.op
```

Print the voltages, currents, and transistor bias conditions at the DC operating point.

8.3 Device Models

Most of the examples in Section 8.2 included a file containing transistor models. SPICE provides a wide variety of MOS transistor models with various trade-offs between complexity and accuracy. Level 1 and Level 3 models were historically important, but they are no longer adequate to accurately model very small modern transistors. BSIM models are more

accurate and are presently the most widely used. Some companies use their own proprietary models. This section briefly describes the main features of each of these models. It also describes how to model diffusion capacitance and how to run simulations in various process corners. The model descriptions are intended only as an overview of the capabilities and limitations of the models; refer to a SPICE manual for a much more detailed description if one is necessary.

8.3.1 Level 1 Models

The SPICE Level 1, or Shichman-Hodges Model [Shichman68] is closely related to the Shockley model described in EQ (2.10), enhanced with channel length modulation and the body effect. The basic current model is:

$$I_{ds} = \begin{cases} 0 & V_{gs} < V_t \quad \text{cutoff} \\ \text{KP}\dfrac{W_{\text{eff}}}{L_{\text{eff}}}\left(1+\text{LAMBDA}\times V_{ds}\right)\left(V_{gs}-V_t-V_{ds}/2\right)V_{ds} & V_{ds} < V_{gs}-V_t \quad \text{linear} \\ \dfrac{\text{KP}}{2}\dfrac{W_{\text{eff}}}{L_{\text{eff}}}\left(1+\text{LAMBDA}\times V_{ds}\right)\left(V_{gs}-V_t\right)^2 & V_{ds} > V_{gs}-V_t \quad \text{saturation} \end{cases} \tag{8.1}$$

The parameters from the SPICE model are given in ALL CAPS. Notice that β is written instead as $\text{KP}(W_{\text{eff}}/L_{\text{eff}})$, where KP is a model parameter playing the role of k' from EQ (2.7). W_{eff} and L_{eff} are the effective width and length, as described in EQ (2.48). The LAMBDA term (LAMBDA = $1/V_A$) models channel length modulation (see Section 2.4.2).

The threshold voltage is modulated by the source-to-body voltage V_{sb} through the body effect (see Section 2.4.3.1). For nonnegative V_{sb}, the threshold voltage is

$$V_t = \text{VTO} + \text{GAMMA}\left(\sqrt{\text{PHI}+V_{sb}} - \sqrt{\text{PHI}}\right) \tag{8.2}$$

Notice that this is identical to EQ (2.30), where VTO is the "zero-bias" threshold voltage V_{t0}, GAMMA is the body effect coefficient γ, and PHI is the surface potential ϕ_s.

The gate capacitance is calculated from the oxide thickness TOX. The default gate capacitance model in HSPICE is adequate for finding the transient response of digital circuits. More elaborate models exist that capture nonreciprocal effects that are important for analog design.

Level 1 models are useful for teaching because they are easy to correlate with hand analysis, but are too simplistic for modern design. Figure 8.12 gives an example of a Level 1 model illustrating the syntax. The model also includes terms to compute the diffusion capacitance, as described in Section 8.3.4.

```
.model NMOS NMOS (LEVEL=1 TOX=40e-10 KP=155E-6 LAMBDA=0.2
+                 VTO=0.4 PHI=0.93 GAMMA=0.6
+                 CJ=9.8E-5 PB=0.72 MJ=0.36
+                 CJSW=2.2E-10 PHP=7.5 MJSW=0.1)
```

FIGURE 8.12 Sample Level 1 Model

8.3.2 Level 2 and 3 Models

The SPICE Level 2 and 3 models add effects of velocity saturation, mobility degradation, subthreshold conduction, and drain-induced barrier lowering. The Level 2 model is based on the Grove-Frohman equations [Frohman69], while the Level 3 model is based on empirical equations that provide similar accuracy, faster simulation times, and better convergence. However, these models still do not provide good fits to the measured I-V characteristics of modern transistors.

8.3.3 BSIM Models

The Berkeley Short-Channel IGFET[1] Model (BSIM) is a very elaborate model that is now widely used in circuit simulation. The models are derived from the underlying device physics but use an enormous number of parameters to fit the behavior of modern transistors. BSIM versions 1, 2, 3v3, and 4 are implemented as SPICE levels 13, 39, 49, and 54, respectively.

BSIM 3 and 4 require entire books [Cheng99, Dunga07] to describe the models. They include over 100 parameters and the device equations span 27 pages. BSIM is quite good for digital circuit simulation. Features of the model include:

- Continuous and differentiable I-V characteristics across subthreshold, linear, and saturation regions for good convergence
- Sensitivity of parameters such as V_t to transistor length and width
- Detailed threshold voltage model including body effect and drain-induced barrier lowering
- Velocity saturation, mobility degradation, and other short-channel effects
- Multiple gate capacitance models
- Diffusion capacitance and resistance models
- Gate leakage models (in BSIM 4)

Some device parameters such as threshold voltage change significantly with device dimensions. BSIM models can be *binned* with different models covering different ranges of length and width specified by LMIN, LMAX, WMIN, and WMAX parameters. For example, one model might cover transistors with channel lengths from 0.18–0.25 μm, another from 0.25–0.5 μm, and a third from 0.5–5 μm. SPICE will complain if a transistor does not fit in one of the bins.

As the BSIM models are so complicated, it is impractical to derive closed-form equations for propagation delay, switching threshold, noise margins, etc., from the underlying equations. However, it is not difficult to find these properties through circuit simulation. Section 8.4 will show simple simulations to plot the device characteristics over the regions of operation that are interesting to most digital designers and to extract effective capacitance and resistance averaged across the switching transition. The simple RC model continues to give the designer important insight about the characteristics of logic gates.

8.3.4 Diffusion Capacitance Models

The p–n junction between the source or drain diffusion and the body forms a diode. We have seen that the diffusion capacitance determines the parasitic delay of a gate and

[1] IGFET in turn stands for Insulated-Gate Field Effect Transistor, a synonym for MOSFET.

depends on the area and perimeter of the diffusion. HSPICE provides a number of methods to specify this geometry, controlled by the ACM (Area Calculation Method) parameter, which is part of the transistor model. The model must also have values for junction and sidewall diffusion capacitance, as described in Section 2.3.3. The diffusion capacitance model is common across most device models including Levels 1–3 and BSIM.

By default, HSPICE models use ACM = 0. In this method, the designer must specify the area and perimeter of the source and drain of each transistor. For example, the dimensions of each diffusion region from Figure 2.8 are listed in Table 8.3 (in units of λ^2 for area or λ for perimeter). A SPICE description of the shared contacted diffusion case is shown in Figure 8.13, assuming `.option scale` is set to the value of λ.

TABLE 8.3 Diffusion area and perimeter

	AS1 / AD2	PS1 / PD2	AD1 / AS2	PD1 / PS2
(a) Isolated contacted diffusion	$W \times 5$	$2 \times W + 10$	$W \times 5$	$2 \times W + 10$
(b) Shared contacted diffusion	$W \times 5$	$2 \times W + 10$	$W \times 3$	$W + 6$
(c) Merged uncontacted diffusion	$W \times 5$	$2 \times W + 10$	$W \times 1.5$	$W + 3$

```
* Shared contacted diffusion
M1   mid   b   bot   gnd   NMOS   W='w'   L=2
+ AS='w*5' PS='2*w+10' AD='w*3' PD='w+6'
M2   top   a   mid   gnd   NMOS   W='w'   L=2
+ AS='w*3' PS='w+6' AD='w*5' PD='2*w+10'
```

FIGURE 8.13 SPICE model of transistors with shared contacted diffusion

The SPICE models also should contain parameters CJ, CJSW, PB, PHP, MJ, and MJSW. Assuming the diffusion is reverse-biased and the area and perimeter are specified, the diffusion capacitance between source and body is computed as described in Section 2.3.3.

$$C_{sb} = \mathrm{AS} \times \mathrm{CJ} \times \left(1 + \frac{V_{sb}}{\mathrm{PB}}\right)^{-MJ} + \mathrm{PS} \times \mathrm{CJSW} \times \left(1 + \frac{V_{sb}}{\mathrm{PHP}}\right)^{-MJSW} \tag{8.3}$$

The drain equations are analogous, with S replaced by D in the model parameters.

The BSIM3 models offer a similar area calculation model (ACM = 10) that takes into account the different sidewall capacitance on the edge adjacent to the gate. Note that the PHP parameter is renamed to PBSW to be more consistent.

$$C_{sb} = \mathrm{AS} \times \mathrm{CJ} \times \left(1 + \frac{V_{sb}}{\mathrm{PB}}\right)^{-MJ} + (\mathrm{PS} - W) \times \mathrm{CJSW} \times \left(1 + \frac{V_{sb}}{\mathrm{PBSW}}\right)^{-MJSW} + W \times \mathrm{CJSWG} \times \left(1 + \frac{V_{sb}}{\mathrm{PBSWG}}\right)^{-MJSWG} \tag{8.4}$$

If the area and perimeter are not specified, they default to 0 in ACM = 0 or 10, grossly underestimating the parasitic delay of the gate. HSPICE also supports ACM = 1, 2, 3, and 12 that provide nonzero default values when the area and perimeter are not specified. Check your models and read the HSPICE documentation carefully.

The diffusion area and perimeter are also used to compute the junction leakage current. However, this current is generally negligible compared to subthreshold leakage in modern devices.

8.3.5 Design Corners

Engineers often simulate circuits in multiple design corners to verify operation across variations in device characteristics and environment. HSPICE includes the `.lib` statement that makes changing libraries easy. For example, the deck in Figure 8.14 runs three simulations on the step response of an unloaded inverter in the TT, FF, and SS corners.

```
* corner.sp
* Step response of unloaded inverter across process corners

*-------------------------------------------------------------------------
* Parameters and models
*-------------------------------------------------------------------------
.option scale=25n
.param SUP=1.0 * Must set before calling .lib
.lib '../models/ibm065/opconditions.lib' TT
.option post

*-------------------------------------------------------------------------
* Simulation netlist
*-------------------------------------------------------------------------
Vdd   vdd   gnd   'SUPPLY'
Vin   a     gnd   PULSE    0 'SUPPLY' 25ps 0ps 0ps 35ps 80ps
M1    y     a     gnd      gnd     NMOS   W=4   L=2
+ AS=20 PS=18 AD=20 PD=18
M2    y     a     vdd      vdd     PMOS   W=8   L=2
+ AS=40 PS=26 AD=40 PD=26

*-------------------------------------------------------------------------
* Stimulus
*-------------------------------------------------------------------------
.tran 0.1ps 80ps
.alter
.lib '../models/ibm065/opconditions.lib' FF
.alter
.lib '../models/ibm065/opconditions.lib' SS
.end
```

FIGURE 8.14 CORNER SPICE deck

The deck first sets `SUP` to the nominal supply voltage of 1.0 V. It then invokes `.lib` to read in the library specifying the TT conditions. In the stimulus, the `.alter` statement is used to repeat the simulation with changes. In this case, the design corner is changed. Altogether, three simulations are performed and three sets of waveforms are generated for the three design corners.

The library file is given in Figure 8.15. Depending on what library was specified, the temperature is set (in degrees Celsius, with `.temp`) and the V_{DD} value `SUPPLY` is calculated from the nominal `SUP`. The library loads the appropriate nMOS and pMOS transistor models. A fast process file might have lower nominal threshold voltages V_{t0}, greater lateral diffusion L_D, and lower diffusion capacitance values.

```
* opconditions.lib
* For IBM 65 nm process

* TT: Typical nMOS, pMOS, voltage, temperature
.lib TT
.temp 70
.param SUPPLY='SUP'
.include 'modelsTT.sp'
.endl TT

* SS: Slow nMOS, pMOS, low voltage, high temperature
.lib SS
.temp 125
.param SUPPLY='0.9 * SUP'
.include 'modelsSS.sp'
.endl SS

* FF: Fast nMOS, pMOS, high voltage, low temperature
.lib FF
.temp 0
.param SUPPLY='1.1 * SUP'
.include 'modelsFF.sp'
.endl FF

* FS: Fast nMOS, Slow pMOS, typical voltage and temperature
.lib FS
.temp 70
.param SUPPLY='SUP'
.include 'modelsFS.sp'
.endl FS

* SF: Slow nMOS, Fast pMOS, typical voltage and temperature
.lib SF
.temp 70
.param SUPPLY='SUP'
.include 'modelsSF.sp'
.endl SF
```

FIGURE 8.15 OPCONDITIONS library

8.4 Device Characterization

Modern SPICE models have so many parameters that the designer cannot easily read key performance characteristics from the model files. A more convenient approach is to run a set of simulations to extract the effective resistance and capacitance, the fanout-of-4 inverter delay, the I-V characteristics, and other interesting data. This section describes these simulations and compares the results across a variety of CMOS processes.

8.4.1 I-V Characteristics

When familiarizing yourself with a new process, a starting point is to plot the current-voltage (I-V) characteristics. Although digital designers seldom make calculations directly from these plots, it is helpful to know the ON current of nMOS and pMOS transistors, how severely velocity-saturated the process is, how the current rolls off below threshold, how the devices are affected by DIBL and body effect, and so forth. These plots are made

with DC sweeps, as discussed in Section 8.2.2. Each transistor is 1 μm wide in a representative 65 nm process at 70 °C with $V_{DD} = 1.0$ V. Figure 8.16 shows nMOS characteristics and Figure 8.17 shows pMOS characteristics.

Figure 8.16(a) plots I_{ds} vs. V_{ds} at various values of V_{gs}, as was done in Figure 8.5. The saturation current would ideally increase quadratically with $V_{gs} - V_t$, but in this plot it shows closer to a linear dependence, indicating that the nMOS transistor is severely velocity-saturated (α closer to 1 than 2 in the α-power model). The significant increase in saturation current with V_{ds} is caused by channel-length modulation. Figure 8.16(b) makes a similar plot for a device with a drawn channel length of twice minimum. The current drops by less than a factor of two because it experiences less velocity saturation. The current is slightly flatter in saturation because channel-length modulation has less impact at longer channel lengths.

Figure 8.16(c) plots I_{ds} vs. V_{gs} on a semilogarithmic scale for $V_{ds} = 0.1$ V and 1.0 V. The straight line at low V_{gs} indicates that the current rolls off exponentially below threshold. The difference in subthreshold leakage at the varying drain voltage reflects the effects

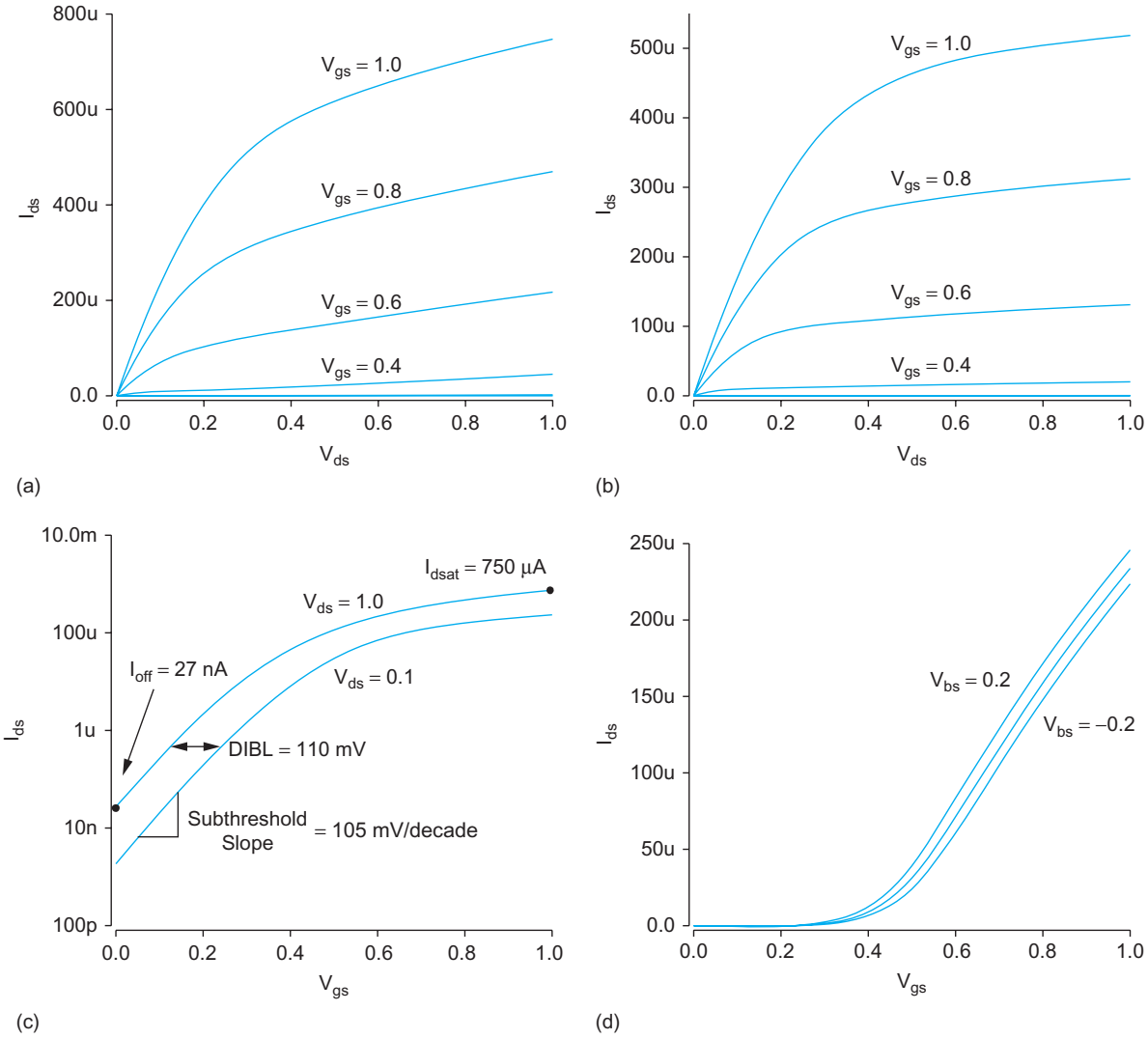

FIGURE 8.16 65 nm nMOS I-V characteristics

of drain-induced barrier lowering (DIBL) effectively reducing V_t at high V_{ds}. The saturation current I_{dsat} is measured at $V_{gs} = V_{ds} = V_{DD}$, while the OFF current I_{off} is measured at $V_{gs} = 0$ and $V_{ds} = V_{DD}$. The subthreshold slope is 105 mV/decade and DIBL reduces the effective threshold voltage by about 110 mV over the range of V_{ds}. The ratio of ON to OFF current is 4–5 orders of magnitude.

Figure 8.16(d) makes a similar plot on a linear scale for $V_{bs} = -0.2$, 0, and 0.2 V. V_{ds} is held constant at 0.1 V. The curves shift horizontally, indicating that the body effect increases the threshold voltage by 125 mV / V as V_{bs} becomes more negative.

Compare the pMOS characteristics in Figure 8.17. The saturation current for a pMOS transistor is lower than for the nMOS (note the different vertical scales), but the device is not as velocity-saturated.

Also compare the 180 nm nMOS characteristics in Figure 8.18. The saturation current is lower in the older technology, leading to lower performance. However, the device characteristics are closer to ideal. The channel-length modulation effect is not as pronounced, though velocity saturation is still severe. The subthreshold slope is 90 nV per decade and DIBL reduces the effective threshold voltage by 40 mV. The ratio of ON to OFF current is 6–7 orders of magnitude.

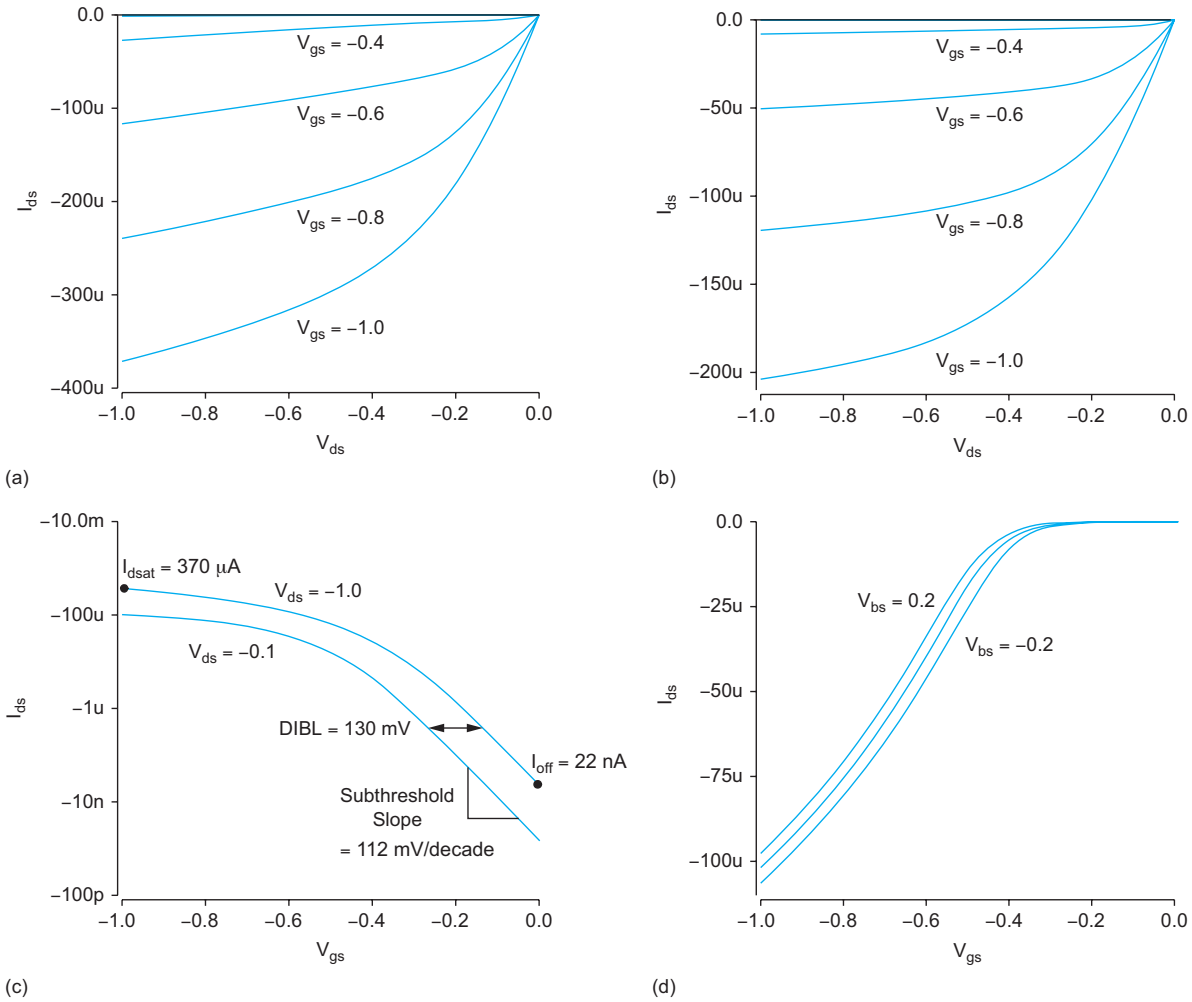

FIGURE 8.17 65 nm pMOS I-V characteristics

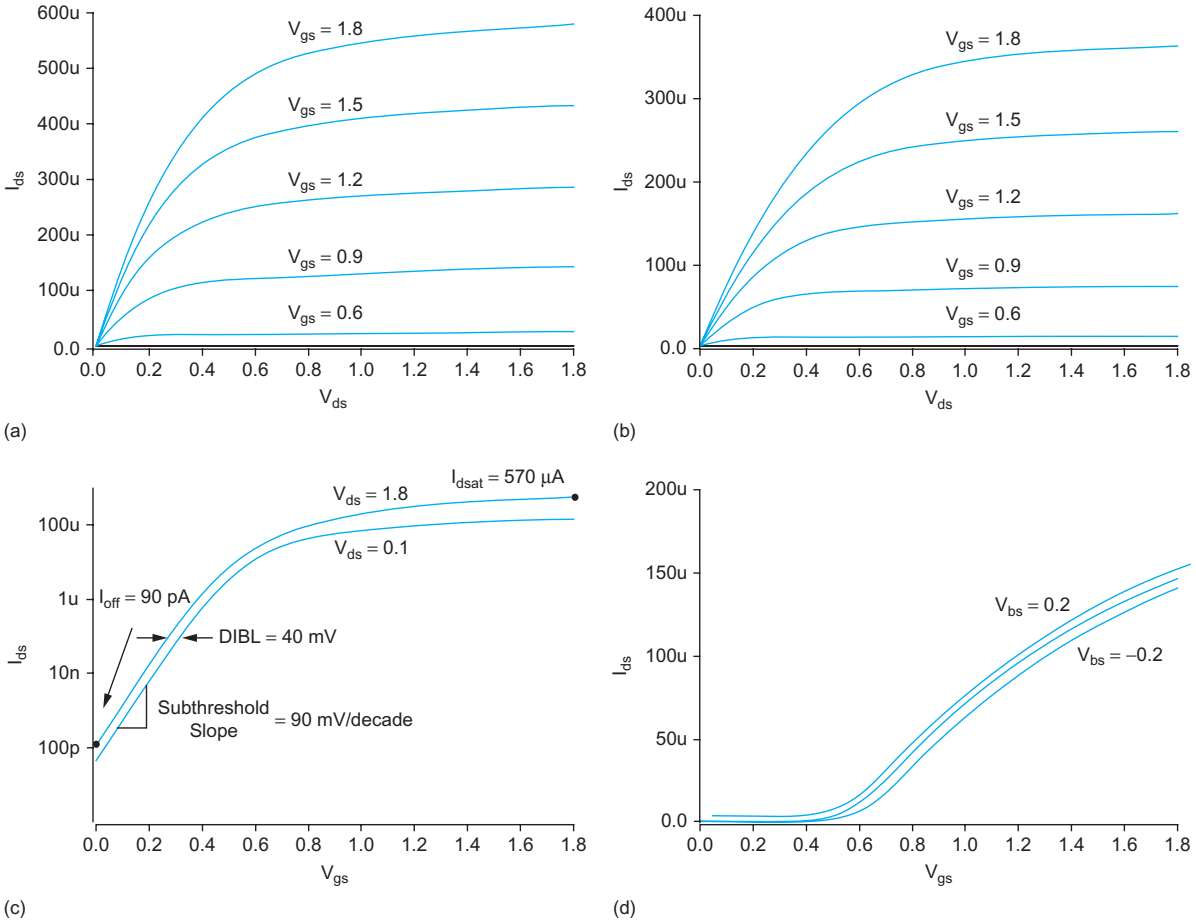

FIGURE 8.18 180 nm nMOS I-V characteristics

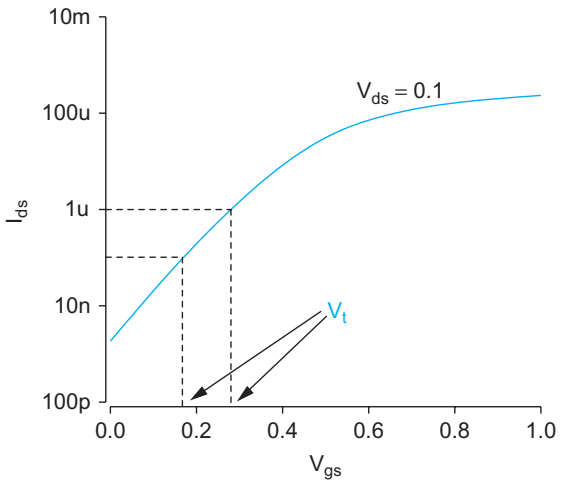

FIGURE 8.19 Constant current threshold voltage extraction method

8.4.2 Threshold Voltage

In the Shockley model, the threshold voltage V_t is defined as the value of V_{gs} below which I_{ds} becomes 0. In the real transistor characteristics shown in Figure 8.16(c), subthreshold current continues to flow for $V_{gs} < V_t$, so measuring or even defining the threshold voltage becomes problematic. Moreover, the threshold voltage varies with L, W, V_{ds}, and V_{bs}. At least eleven different methods have been used in the literature to determine the threshold voltage from measured I_{ds}-V_{gs} data [Ortiz-Conde02]. This section will explore two common methods (constant current and linear extrapolation) and a hybrid that combines the advantages of each.

The *constant current* method defines threshold as the gate voltage at a given drain current I_{crit}. This method is easy to use, but depends on an arbitrary choice of critical drain current. A typical choice of I_{crit} is 0.1 μA × (W/L). Figure 8.19 shows how the extracted threshold voltage varies with the choice of I_{crit} = 0.1 or 1 μA at V_{ds} = 100 mV.

The *linear extrapolation* (or *maximum-g_m*) method extrapolates the gate voltage from the point of maximum slope on the I_{ds}-V_{gs} characteristics. It is unambiguous but valid only for the linear region of operation (low V_{ds}) because of the series resistance of the source/drain diffusion and because drain-induced barrier lowering effectively reduces the threshold at high V_{ds}. Figure 8.20 shows how the threshold is extracted from measured data using the linear extrapolation method at V_{ds} = 100 mV. Observe that this method can give a significantly different threshold voltage and nonnegligible current at threshold, so it is important to check how the threshold voltage was measured when interpreting threshold voltage specifications. I_{crit} is defined to be the value of I_{ds} at $V_{gs} = V_t$.

[Zhou99] describes a hybrid method of extracting threshold voltage that is valid for all values of V_{ds} and does not depend on an arbitrary choice of critical current. V_t and I_{crit} are found at low V_{ds} (e.g., 100 mV) for a given value of L and W using the linear extrapolation method. For other values of V_{ds}, V_t is defined to be the gate voltage when $I_{ds} = I_{crit}$.

Figure 8.21(a) plots the threshold voltage V_t vs. length for a 16 λ wide device over a variety of design corners and temperatures. The threshold is extracted using the linear extrapolation method and clearly is not constant. It decreases with temperature and is lower in the FF corner than in the SS corner. In an ideal long-channel transistor, the threshold is independent of width and length. In a real device, the geometry sensitivity depends on the particular doping profile of the process. This data shows the threshold decreasing with L, but in many processes, the threshold increases with L. Figure 8.21(b) plots V_t against V_{ds} for 16/2 λ transistors using Zhou's method. The threshold voltage decreases with V_{ds} because of DIBL.

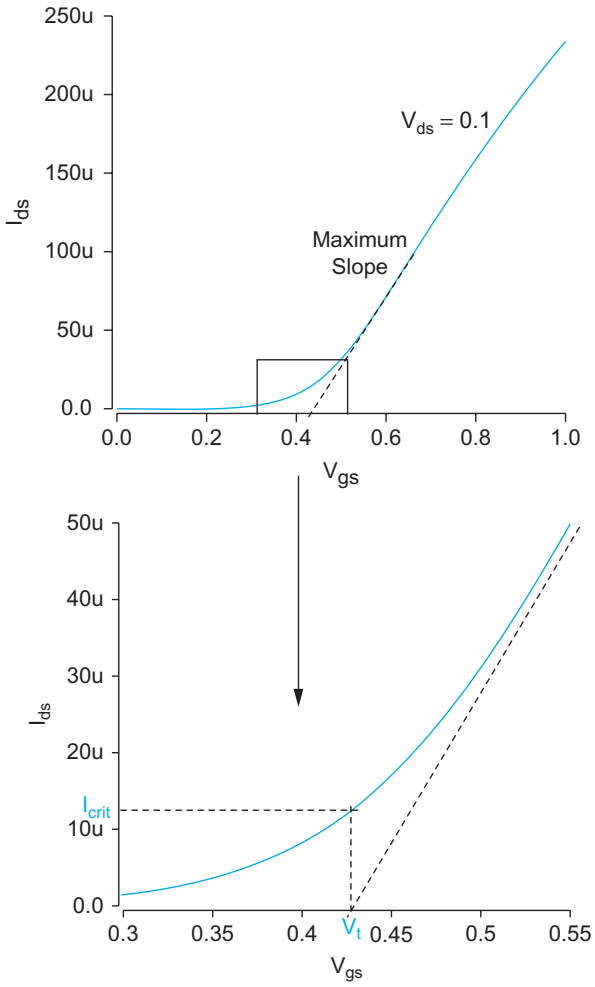

FIGURE 8.20 Linear extrapolation threshold voltage extraction method

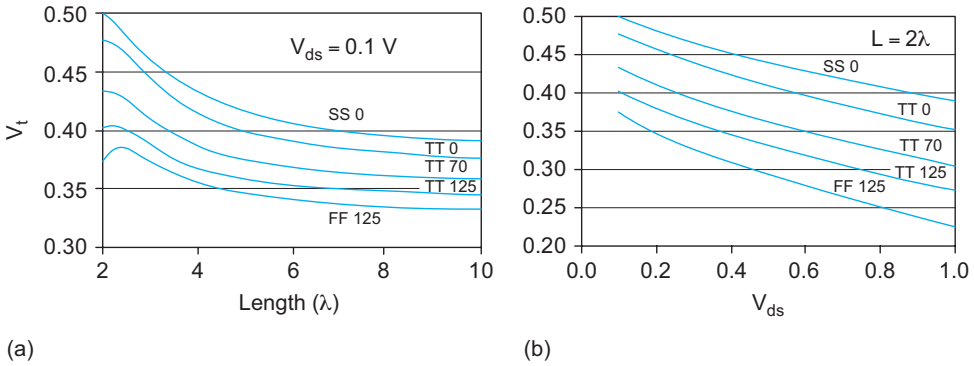

FIGURE 8.21 MOS threshold voltages

The lesson is that V_t depends on length, width, temperature, processing, and how you define it. The current does not abruptly drop to zero at threshold and is significant even for OFF devices in nanometer processes.

8.4.3 Gate Capacitance

When using RC models to estimate gate delay, we need to know the effective gate capacitance for delay purposes. In Section 2.3.2, we saw that the gate capacitance is voltage-dependent. The gate-to-drain component may be effectively doubled when a gate switches because the gate and drain switch in opposite directions. Nevertheless, we can obtain an effective capacitance averaged across the switching time. We use fanout-of-4 inverters to represent gates with "typical" switching times because we know from logical effort that circuits perform well when the stage effort is approximately 4.

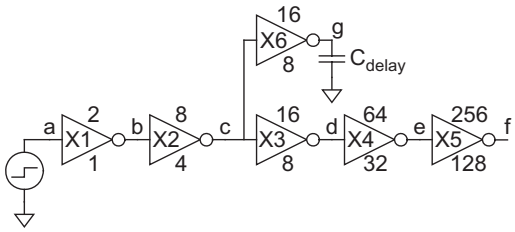

FIGURE 8.22 Circuit for extracting effective gate capacitance for delay estimation

Figure 8.22 shows a circuit for determining the effective gate capacitance of inverter $X4$. The approach is to adjust the capacitance C_{delay} until the average delay from `c` to `g` equals the delay from `c` to `d`. Because $X6$ and $X3$ have the same input slope and are the same size, when they have the same delay, C_{delay} must equal the effective gate capacitance of $X4$. $X1$ and $X2$ are used to produce a reasonable input slope on node `c`. A single inverter could suffice, but the inverter pair is even better because it provides a slope on `c` that is essentially independent of the rise time at `a`. $X5$ is the load on $X4$ to prevent node `e` from switching excessively fast, which would overpredict the significance of the gate-to-drain capacitance in $X4$.

Figure 8.23 (on page 309) lists a SPICE deck that uses the optimizer to automatically tune C_{delay} until the delays are equalized. This capacitance is divided by the total gate width (in μm) of $X4$ to obtain the capacitance per micron of gate width $C_{permicron}$. This capacitance is listed as C_g (delay) in Table 8.5 for a variety of processes. Note that the deck sets diffusion area and perimeter to 0 to measure only the gate capacitance.

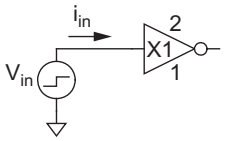

FIGURE 8.24 Circuit for extracting effective gate capacitance for power estimation

Gate capacitance is also important for dynamic power consumption, as was given in EQ (5.10). The effective gate capacitance for power is typically somewhat higher than for delay because C_{gd} is effectively doubled by the Miller effect when we wait long enough for the drain to completely switch. Figure 8.24 shows a circuit for measuring gate capacitance for power purposes. A voltage step is applied to the input, and the current out of the voltage source is integrated. The effective capacitance for dynamic power consumption is:

$$C_{\text{eff}-\text{power}} = \frac{\int i_{\text{in}}(t)\,dt}{V_{DD}} \tag{8.5}$$

Again, this capacitance can be divided by the total transistor width to find the effective gate capacitance per micron.

8.4.4 Parasitic Capacitance

The parasitic capacitance associated with the source or drain of a transistor includes the gate-to-diffusion overlap capacitance, C_{gol}, and the diffusion area and perimeter capacitance C_{jb} and C_{jbsw}. As discussed in Section 8.3.4, some models assign a different capacitance C_{jbswg} to the perimeter along the gate side. The diffusion capacitance is voltage-dependent, but as with gate capacitance, we can extract an effective capacitance averaged over the switching transition to use for delay estimation.

```
* capdelay.hsp
* Extract effective gate capacitance for delay estimation.
*-------------------------------------------------------------------------
* Parameters and models
*-------------------------------------------------------------------------
.option scale=25n
.param SUP=1.0 * Must set before calling .lib
.lib '../models/ibm065/opconditions.lib' TT
.option post
*-------------------------------------------------------------------------
* Subcircuits
*-------------------------------------------------------------------------
.global  vdd   gnd
.subckt  inv   a     y
M1       y     a     gnd   gnd   NMOS   W=16 L=2 AD=0 AS=0 PD=0 PS=0
M2       y     a     vdd   vdd   PMOS   W=32 L=2 AD=0 AS=0 PD=0 PS=0
.ends
*-------------------------------------------------------------------------
* Simulation netlist
*-------------------------------------------------------------------------
Vdd      vdd   gnd   'SUPPLY'      * SUPPLY is set by .lib call
Vin      a     gnd   pulse 0 'SUPPLY' 0ps 20ps 20ps 120ps 280ps
X1       a     b     inv           * set appropriate slope
X2       b     c     inv   M=4     * set appropriate slope
X3       c     d     inv   M=8     * drive real load
X4       d     e     inv   M=32    * real load
X5       e     f     inv   M=128   * load on load (important!)
X6       c     g     inv   M=8     * drive linear capacitor
cdelay   g     gnd   'CperMicron*32*(16+32)*25n/1u' * linear capacitor
*-------------------------------------------------------------------------
* Optimization setup
*-------------------------------------------------------------------------
.measure errorR param='invR - capR' goal=0
.measure errorF param='invF - capF' goal=0
.param CperMicron=optrange(2f, 0.2f, 3.0f)
.model optmod opt itropt=30
.measure CperMic param = 'CperMicron'
*-------------------------------------------------------------------------
* Stimulus
*-------------------------------------------------------------------------
.tran 1ps 280ns SWEEP OPTIMIZE = optrange
+       RESULTS=errorR,errorF MODEL=optmod
.measure invR
+       TRIG v(c)  VAL='SUPPLY/2' FALL=1
+       TARG v(d)  VAL='SUPPLY/2' RISE=1
.measure  capR
+       TRIG v(c)  VAL='SUPPLY/2' FALL=1
+       TARG v(g)  VAL='SUPPLY/2' RISE=1
.measure invF
+       TRIG v(c)  VAL='SUPPLY/2' RISE=1
+       TARG v(d)  VAL='SUPPLY/2' FALL=1
.measure  capF
+       TRIG v(c)  VAL='SUPPLY/2' RISE=1
+       TARG v(g)  VAL='SUPPLY/2' FALL=1
.end
```

FIGURE 8.23 CAPDELAY SPICE deck

Figure 8.25 shows circuits for extracting these capacitances. They operate in much the same way as the gate capacitance extraction from Section 8.4.3. The first two fanout-of-4 inverters shape the input slope to match a typical gate. *X*3 drives the drain of an OFF

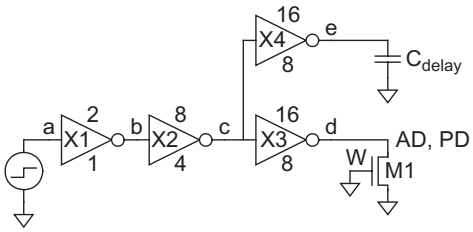

FIGURE 8.25 Circuit for extracting effective parasitic capacitance for delay estimation

transistor *M*1 with specified *W*, AD, and PD. *X*4 drives a simple capacitor, whose value is optimized so that the delay of *X*3 and *X*4 are equal. This value is the effective capacitance of *M*1's drain. Similar simulations must be run to find the parasitic capacitances of pMOS transistors.

Table 8.4 lists the appropriate values of *W*, AD, and PD to extract each of the capacitances. The sizes are chosen such that the gate delays and slope on node d are reasonable when a unit transistor is 16 λ wide (as in Figure 8.23). It also gives values to find the effective capacitance C_d of isolated-contacted, shared-contacted, and merged-uncontacted diffusion regions. The capacitance is found, assuming the transistors are wide enough that the perimeter perpendicular to the polysilicon gate is a negligible fraction of the overall capacitance. The AD and PD dimensions are based on the layouts of Figure 2.8; you should substitute your own design rules. The total capacitance of shared and merged regions should be split between the two transistors sharing the diffusion node. The capacitance can be converted to units per micron (or per micron squared) by normalizing for the value of λ. For example, in our 65 nm process, if C_{delay} is 23 fF for gate overlap, the capacitance per micron is

$$C_{gol} = \frac{23\ \text{fF}}{(1600\ \lambda)\left(\frac{0.025\ \mu\text{m}}{\lambda}\right)} = 0.57\frac{\text{fF}}{\mu\text{m}} \tag{8.6}$$

TABLE 8.4 Dimensions for diffusion capacitance extraction

	W (λ)	AD (λ^2)	PD (λ)	To find effective *C* per micron
C_{gol}	1600	0	0	$C_{\text{delay}}/1600\lambda$ (per μm)
C_{jb}	0	8000	0	$C_{\text{delay}}/8000\lambda^2$ (per μm^2)
C_{jbsw}	0	0	1600	$C_{\text{delay}}/1600\lambda$ (per μm)
C_{jbswg}	1600	0	1600	$C_{\text{delay}}/1600\lambda - C_{gol}$ (per μm)
C_d (isolated-contacted)	1600	8000	3200	$C_{\text{delay}}/1600\lambda$ (per μm of gate width)
C_d (shared-contacted)	3200	9600	3200	$C_{\text{delay}}/1600\lambda$ (per μm of gate width)
C_d (merged-uncontacted)	3200	4800	3200	$C_{\text{delay}}/1600\lambda$ (per μm of gate width)

8.4.5 Effective Resistance

If a unit transistor has gate capacitance *C*, parasitic capacitance C_d, and resistance R_n (for nMOS) or R_p (for pMOS), the rising and falling delays of a fanout-of-*h* inverter with a 2:1 *P*/*N* ratio can be found according to Figure 8.26. These delays can readily be measured from the FO4 inverter simulation in Figure 8.10 by changing *h*.

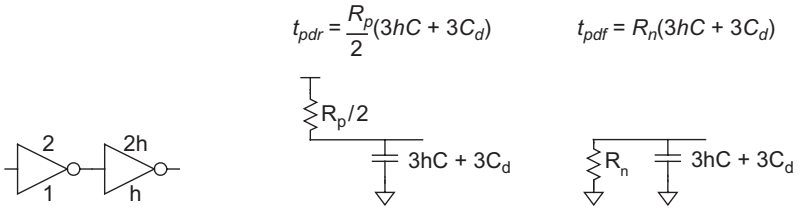

FIGURE 8.26 RC delay model for fanout-of-*h* inverter

The dependence on parasitics can be removed by calculating the difference between delays at different fanouts. For example, the difference between delays for $h = 3$ and $h = 4$ are

$$\Delta t_{pdr} = \frac{R_p}{2}\left(3 \times 4 \times C + 3C_d\right) - \frac{R_p}{2}\left(3 \times 3 \times C + 3C_d\right) = \tfrac{3}{2} R_p C$$
$$\Delta t_{pdf} = R_n\left(3 \times 4 \times C + 3C_d\right) - R_n\left(3 \times 3 \times C + 3C_d\right) = 3R_n C \tag{8.7}$$

As C is known from the effective gate capacitance extraction, R_n and R_p are readily calculated. These represent the effective resistance of single nMOS and pMOS transistors for delay estimation.

When two unit transistors are in series, each nominally would have the same effective resistance, giving twice the overall resistance. However, in modern processes where the transistors usually experience some velocity saturation, each transistor sees a smaller V_{ds} and hence less velocity saturation and a lower effective resistance. We can determine this resistance by simulating fanout-of-h tristates in place of inverters, as shown in Figure 8.27. By a similar reasoning, the difference between delays from c to d for $h = 2$ and $h = 3$ is

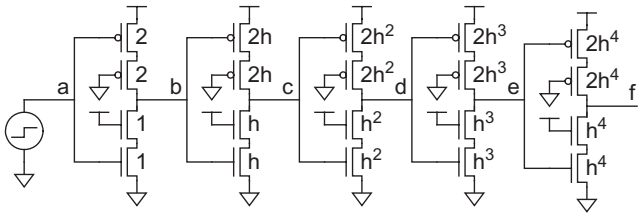

FIGURE 8.27 Circuit for extracting effective series resistance

$$\Delta t_{pdr} = \tfrac{3}{2}\left(2R_{p\text{-series}}\right)C$$
$$\Delta t_{pdf} = 3\left(2R_{n\text{-series}}\right)C \tag{8.8}$$

As C is still known, we can extract the effective resistance of series nMOS and pMOS transistors for delay estimation and should expect this resistance to be smaller than for single transistors.

It is important to use realistic input slopes when extracting effective resistance because the delay varies with input slope. *Realistic* means that the input and output edge rates should be comparable; if a step input is applied, the output will transition faster and the effective resistance will appear to decrease. h was chosen in this section to give stage efforts close to 4.

8.4.6 Comparison of Processes

Table 8.5 compares the characteristics of a variety of CMOS processes with feature sizes ranging from 2 μm down to 65 nm. The older models are obtained from MOSIS wafer test results [Piña02], while the newer models are from IBM or TSMC. The MOSIS models use ACM = 0, so the diffusion sidewall capacitance is treated the same along the gate and the other walls. The 0.6 μm process operates at either $V_{DD} = 5$ V (for higher speed) or $V_{DD} = 3.3$ V (for lower power). All characteristics are extracted for TTTT conditions (70 °C) for normal-V_t transistors.

Transistor lengths are usually shorter than the nominal feature size. For example, in the 0.6 μm process, MOSIS preshrinks polysilicon by 0.1 μm before generating masks. In the IBM process, transistors are drawn somewhat shorter than the feature size. Moreover, gates are usually processed such that the effective channel length is even shorter than the drawn channel length. The shorter channels make transistors faster than one might expect simply based on feature size.

TABLE 8.5 Device characteristics for a variety of processes

Vendor		Orbit	HP	AMI	AMI	TSMC	TSMC	TSMC	IBM	IBM	IBM
Model		MOSIS	MOSIS	MOSIS	MOSIS	MOSIS	MOSIS	TSMC	IBM	IBM	IBM
Feature Size f	nm	2000	800	600	600	350	250	180	130	90	65
V_{DD}	V	5	5	5	3.3	3.3	2.5	1.8	1.2	1.0	1.0
Gates											
C_g (delay)	fF/μm	1.77	1.67	1.55	1.48	1.90	2.30	1.67	1.04	0.97	0.80
C_g (power)	fF/μm	2.24	1.70	1.83	1.76	2.20	2.92	2.06	1.34	1.23	1.07
FO4 Inv. Delay	ps	856	297	230	312	210	153	75.6	45.9	37.2	17.2
nMOS											
C_d (isolated)	fF/μm	·1.19	1.11	1.14	1.21	1.63	1.88	1.12	0.94	0.89	0.76
C_d (shared)	fF/μm	1.62	1.43	1.41	1.50	2.04	2.60	1.62	1.56	1.60	1.28
C_d (merged)	fF/μm	1.48	1.36	1.19	1.24	1.60	2.16	1.41	1.40	1.51	1.20
R_n (single)	k$\Omega \cdot \mu$m	30.3	10.1	9.19	11.9	5.73	4.02	2.69	2.54	2.35	1.34
R_n (series)	k$\Omega \cdot \mu$m	22.1	6.95	6.28	8.59	4.01	3.10	2.00	1.93	1.81	1.13
V_{tn} (const. I)	V	0.65	0.65	0.70	0.70	0.59	0.48	0.41	0.32	0.32	0.31
V_{tn} (linear ext.)	V	0.65	0.75	0.76	0.76	0.67	0.57	0.53	0.43	0.43	0.43
I_{dsat}	μA/μm	152	380	387	216	450	551	566	478	497	755
I_{off}	pA/μm	2.26	9.36	2.21	1.45	6.57	56.3	93.9	1720	4000	33400
I_{gate}	pA/μm	n/a	n/a	n/a	n/a	n/a	n/a	n/a	1.22	3620	8520
pMOS											
C_d (isolated)	fF/μm	1.42	1.17	1.31	1.42	1.89	2.07	1.24	0.94	0.74	0.73
C_d (shared)	fF/μm	1.92	1.62	1.73	1.86	2.37	2.89	1.79	1.56	1.25	1.25
C_d (merged)	fF/μm	1.52	1.23	1.35	1.43	1.83	2.40	1.56	1.41	1.16	1.18
R_p (single)	k$\Omega \cdot \mu$m	67.1	26.7	19.9	29.6	16.1	8.93	6.51	6.39	5.47	2.87
R_p (series)	k$\Omega \cdot \mu$m	53.9	21.4	15.4	23.6	13.3	6.91	5.41	5.48	4.92	2.42
$\lvert V_{tp}\rvert$ (const. I)	V	0.72	0.91	0.90	0.90	0.83	0.46	0.43	0.33	0.35	0.33
$\lvert V_{tp}\rvert$ (linear ext.)	V	0.71	0.94	0.93	0.93	0.88	0.52	0.51	0.42	0.43	0.42
I_{dsat}	μA/μm	70.5	154	215.3	99.0	181	245	228	177	187	360
I_{off}	pA/μm	2.18	1.57	2.08	1.38	2.06	30.1	25.2	1330	2780	19500
I_{gate}	pA/μm	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.06	1210	2770

The gate capacitance for delay held steady near 2 fF/μm for many generations, as scaling theory would predict, but abruptly dropped after the 180 nm generation. The gate capacitance for power is slightly higher than that for delay as discussed in Section 8.4.3.

The FO4 inverter delay has steadily improved with feature size as constant field scaling predicts. It fits our rule from Section 4.4.3 of one third to one half of the effective channel length, when delay is measured in picoseconds and length in nanometers.

Diffusion capacitance of an isolated contacted source or drain has been 1–2 fF/μm for both nMOS and pMOS transistors over many generations. The capacitance of a shared contacted diffusion region is slightly higher because it has more area and includes two gate overlaps. The capacitance of the merged diffusion reflects two gate overlaps but a smaller diffusion area. Half the capacitance of the shared and merged diffusions is allocated to each of the transistors connected to the diffusion region.

The effective resistance of a 1 μm wide transistor has decreased with process scaling in proportion to the feature size f. However, the resistance of a unit (4/2 λ) nMOS transistor, $R/2f$, has remained roughly constant around 8 kΩ, as constant field scaling theory would predict. The effective resistance of pMOS transistors is 2–3 times that of nMOS transistors. A pair of nMOS transistors in series each have lower effective resistance than a single device because each has a smaller V_{ds} and thus experiences less velocity saturation. Series pMOS transistors show less pronounced improvement because they were not as velocity-saturated to begin with.

Threshold voltages are reported at V_{ds} = 100 mV for 16/2 λ devices using both the constant current (at $I_{crit} = 0.1(W/L)$ μA for nMOS and $0.06(W/L)$ for pMOS) and linear extrapolation methods. Threshold voltages have generally decreased, but not as fast as channel length or supply voltage (because of subthreshold leakage). Therefore, the V_{DD}/V_t ratio is decreasing and pass transistor circuits with threshold drops do not perform well in modern processes.

Saturation current per micron has increased somewhat through aggressive device design as feature size decreases even though constant field scaling would suggest it should remain constant. OFF current was on the order of a few picoamperes per micron in old processes, but is exponentially increasing in nanometer processes because of subthreshold conduction through devices with low threshold voltages. The current at threshold using the linear extrapolation method is somewhat higher than the constant current I_{crit}, corresponding to the higher threshold voltages found by the linear extrapolation method. Gate leakage has become significant below 90 nm.

8.4.7 Process and Environmental Sensitivity

Table 8.6 shows how the IBM 65 nm process characteristics vary with process corner, voltage, and temperature. The FO4 inverter delay varies by a factor of two between best and worst case. In the TT process, inverter delay varies by about 0.12%/°C and by about 1% for every percent of supply voltage change. These figures agree well with the Artisan library data from Section 7.2.4. Gate and diffusion capacitance change only slightly with process, but effective resistance is inversely proportional to supply voltage and highly sensitive to temperature and device corners. I_{off} subthreshold leakage rises dramatically at high temperature or in the fast corner where threshold voltages are lower.

8.5 Circuit Characterization

The device characterization techniques from the previous section are typically run once by engineers who are familiarizing themselves with a new process. SPICE is used more often to characterize entire circuits. This section gives some pointers on simulating paths and describes how to find the DC transfer characteristics, logical effort, and power consumption of logic gates.

8.5.1 Path Simulations

The delays of most static CMOS circuit paths today are computed with a static timing analyzer (see Sections 4.6 and 14.4.1.4). As long as the noise sources (particularly coupling and power supply noise) are controlled, the circuits will operate correctly and will

TABLE 8.6 Process corners of IBM 65 nm process

nMOS		T	F	S	F	S	T	T	T	T
pMOS		T	F	S	S	F	T	T	T	T
V_{DD}	V	1.0	1.1	0.9	1.0	1.0	1.1	0.9	1.0	1.0
T	°C	70	0	125	70	70	70	70	0	125
Gates										
C_g (delay)	fF/μm	0.80	0.79	0.80	0.82	0.79	0.82	0.78	0.79	0.81
C_g (power)	fF/μm	1.07	1.05	1.07	1.07	1.04	1.07	1.04	1.07	1.06
FO4 Inv. Delay	ps	17.2	12.2	24.4	17.4	17.1	15.1	20.4	16.6	17.5
nMOS										
C_d (isolated)	fF/μm	0.76	0.72	0.79	0.72	0.80	0.75	0.77	0.75	0.76
C_d (shared)	fF/μm	1.28	1.22	1.33	1.22	1.33	1.26	1.29	1.27	1.28
C_d (merged)	fF/μm	1.20	1.15	1.25	1.15	1.25	1.19	1.22	1.20	1.21
R_n (single)	kΩ · μm	1.34	0.96	1.92	1.21	1.49	1.16	1.63	1.31	1.37
R_n (series)	kΩ · μm	1.13	0.79	1.66	0.96	1.31	0.97	1.39	1.09	1.16
V_{tn} (const. I)	V	0.31	0.32	0.30	0.27	0.34	0.31	0.31	0.36	0.27
V_{tn} (linear ext.)	V	0.43	0.45	0.43	0.40	0.47	0.43	0.43	0.48	0.40
I_{dsat}	μA/μm	755	1094	510	844	672	919	596	793	731
I_{off}	nA/μm	33.4	22.4	38.9	95.0	12.3	41.5	2.7	4.6	120
I_{gate}	nA/μm	8.5	13.6	5.0	8.9	8.1	12.6	5.7	8.1	8.9
pMOS										
C_d (isolated)	fF/μm	0.73	0.69	0.77	0.77	0.70	0.72	0.74	0.72	0.74
C_d (shared)	fF/μm	1.25	1.18	1.32	1.30	1.20	1.23	1.26	1.23	1.26
C_d (merged)	fF/μm	1.18	1.12	1.24	1.23	1.13	1.16	1.20	1.17	1.19
R_p (single)	kΩ · μm	2.87	2.09	3.99	3.10	2.65	2.46	3.47	2.82	2.89
R_p (series)	kΩ · μm	2.42	1.67	3.47	2.69	2.04	1.16	3.05	2.40	2.35
V_{tn} (const. I)	V	0.33	0.36	0.32	0.37	0.30	0.33	0.33	0.39	0.28
V_{tp} (linear ext.)	V	0.42	0.44	0.41	0.45	0.39	0.42	0.42	0.47	0.39
I_{dsat}	μA/μm	360	517	247	319	407	438	285	373	353
I_{off}	nA/μm	19.5	7.4	27.7	7.0	53.0	24.0	15.7	1.4	86.1
I_{gate}	nA/μm	2.8	4.3	1.7	2.6	2.9	4.0	1.9	2.5	2.9

correlate reasonably well with static timing predictions. However, SPICE-level simulation is important for sensitive circuits such as the clock generator and distribution network, custom memory arrays, and novel circuit techniques.

Most experienced designers begin designing paths based on simple models in order to understand what aspects are most important, evaluate design trade-offs, and obtain a qualitative prediction of the results. The ideal Shockley transistor models, RC delay models, and logical effort are all helpful here because they are simple enough to give insight. When a good first-pass design is ready, the designer simulates the circuit to verify that it operates correctly and meets delay and power specifications. Just as few new software programs run correctly before debugging, the simulation often will be incorrect at first. Unless the designer knows what results to expect, it is tempting to trust the false results that are nicely printed with beguilingly many significant figures. Once the circuit appears to be correct, it

should be checked across design corners to verify that it operates in all cases. Section 7.2.4 gives examples of circuits sensitive to various corners.

Simulation is cheap, but silicon revisions are devastating expensive. Therefore, it is important to construct a circuit model that captures all of the relevant conditions, including real input waveforms, appropriate output loading, and adequate interconnect models. When matching is important, you must consider the effects of mismatches that are not given in the corner files (see Section 8.5.5). However, as SPICE decks get more complicated, they run more slowly, accumulate more mistakes, and are more difficult to debug. A good compromise is to start simple and gradually add complexity, ensuring after each step that the results still make sense.

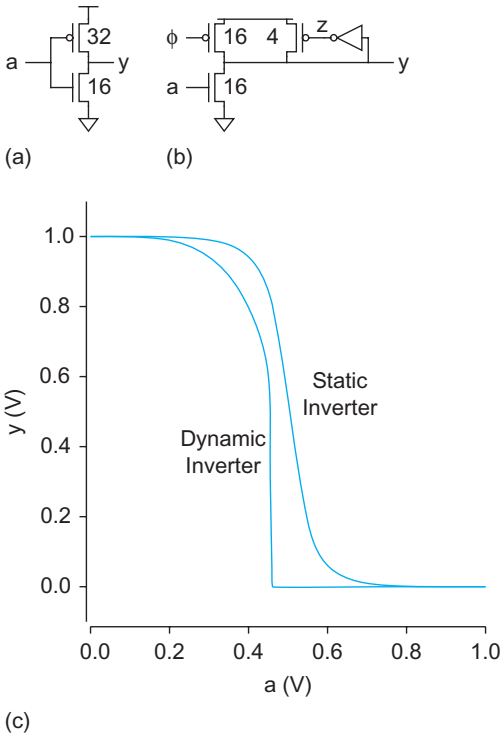

FIGURE 8.28 Circuits for DC transfer analysis

8.5.2 DC Transfer Characteristics

The `.dc` statement is useful for finding the transfer characteristics and noise margins of logic gates. Figure 8.29 shows an example of characterizing static and dynamic inverters (dynamic logic is covered in Section 9.2.4). Figure 8.28(a and b) show the circuit schematics of each gate. Figure 8.28(c) shows the simulation results. The static inverter characteristics are nearly symmetric around $V_{DD}/2$. The dynamic inverter has a lower switching threshold and its output drops abruptly beyond this threshold because positive feedback turns off the keeper.

Note that when the input `a` is 0 and the dynamic inverter is in evaluation ($\phi = 1$), the output would be stable at either 0 or 1. To find the transfer characteristics, we initialize the gate with a 1 output using the `.ic` command.

8.5.3 Logical Effort

The logical effort and parasitic delay of each input of a gate can be measured by fitting a straight line to delay vs. fanout simulation results. As with the FO4 inverter example, it is important to drive the gate with an appropriate input waveform and to provide two stages of loads. Figure 8.30(a) shows an example of a circuit for characterizing the delay of a 2-input NAND gate $X3$ using the `M` parameter to simulate multiple gates in parallel. Figure 8.30(b) plots the delay vs. fanout in a 65 nm process for an inverter and the 2-input NAND. The data is well-fit by a straight line even though the transistors experience all sorts of nonlinear and nonideal effects. This shows that the linear delay model is quite accurate as long as the input and output slopes are consistent.

The `SWEEP` command is convenient to vary the fanout and repeat the transient simulation multiple times. For example, the following statement runs eight simulations varying `H` from 1 to 8 in steps of 1.

```
.tran 1ps 1000ps SWEEP H 1 8 1
```

To characterize an entire library, you can write a script in a language such as Perl or Python that generates the appropriate SPICE decks, invokes the simulator, and post-processes the list files to extract the data and do the curve fit.

Recall that τ is the coefficient of h (i.e., the slope) in a delay vs. fanout plot for an inverter; in this process it is 3.3 ps. The parasitic delay of the inverter is found from the

```
* invdc.sp
* Static and dynamic inverter DC transfer characteristics

*-----------------------------------------------------------------------
* Parameters and models
*-----------------------------------------------------------------------
.param SUPPLY=1.0
.option scale=25n
.include '../models/ibm065/models.sp'
.temp 70
.option post

*-----------------------------------------------------------------------
* Simulation netlist
*-----------------------------------------------------------------------
Vdd   vdd   gnd   'SUPPLY'
Va    a     gnd   0
Vclk  clk   gnd   'SUPPLY'
* Static Inverter
M1    y1    a     gnd   gnd   NMOS   W=16   L=2
M2    y1    a     vdd   vdd   PMOS   W=32   L=2
* Dynamic Inverter
M3    y2    a     gnd   gnd   NMOS   W=16   L=2
M4    y2    clk   vdd   vdd   PMOS   W=16   L=2
M5    y2    z     vdd   vdd   PMOS   W=4    L=2
M6    z     y2    gnd   gnd   NMOS   W=4    L=2
M7    z     y2    vdd   vdd   PMOS   W=8    L=2
.ic V(y2) = 'SUPPLY'

*-----------------------------------------------------------------------
* Stimulus
*-----------------------------------------------------------------------
.dc Va 0 1.0 0.01
.end
```

FIGURE 8.29 INVDC SPICE deck for DC transfer analysis

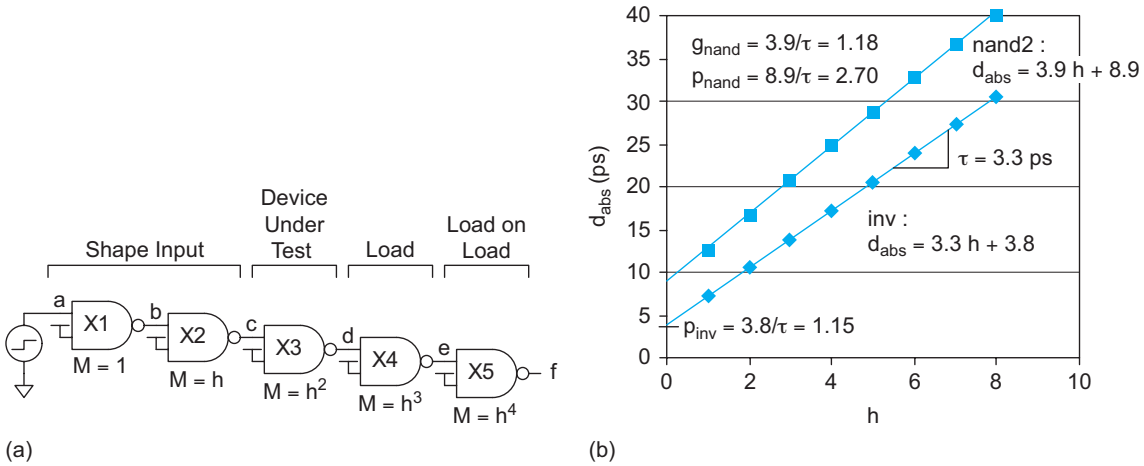

FIGURE 8.30 Logical effort characterization of 2-input NAND gate and inverter

y-intercept of the fit line; it is 3.8 ps, or 1.15 in normalized units. Similarly, the logical effort and parasitic delay of the NAND gate are obtained by normalizing the slope and y-intercept by τ.

Table 8.7 compares the logical effort and parasitic delay of the different inputs of multi-input NAND gates for rising, falling, and average output transitions in the IBM 65 nm process. For rising and falling transitions, we still normalize against the value of τ found from the average delay of an inverter. Input *A* is the outermost (closest to power or ground). As discussed in Section 9.2.1.3, the outer input has higher parasitic delay, but slightly lower logical effort. The rising and falling delays in this process are quite different because pMOS transistors have less than half the mobility of nMOS transistors and because the nMOS transistors are quite velocity-saturated so that series transistors have less resistance than expected.

TABLE 8.7 Logical effort and parasitic delay of different inputs of multi-input NAND gates

# of inputs	Input	Rising Logical Effort g_u	Falling Logical Effort g_u	Average Logical Effort g	Rising Parasitic Delay p_u	Falling Parasitic Delay p_d	Average Parasitic Delay p
2	*A*	1.40	1.12	**1.26**	2.46	2.48	**2.47**
	B	1.31	1.16	**1.24**	1.97	1.82	**1.89**
3	*A*	1.76	1.27	**1.51**	4.77	4.10	**4.44**
	B	1.73	1.32	**1.52**	3.93	3.60	**3.77**
	C	1.59	1.38	**1.48**	3.05	2.43	**2.74**
4	*A*	2.15	1.42	**1.78**	7.63	5.94	**6.79**
	B	2.09	1.48	**1.78**	6.67	5.37	**6.02**
	C	2.08	1.53	**1.80**	5.32	4.51	**4.91**
	D	1.90	1.59	**1.75**	4.04	2.93	**3.49**

Table 8.8 compares the average logical effort and parasitic delay of a variety of gates in many different processes. In each case, the simulations are performed in the TTTT corner for the outer input. For reference, the FO4 inverter delay and τ are given for each process. The logical effort of gates with series transistors is lower than predicted in Section 4.4.1 because one of the transistors is already fully ON and hence has a lower effective resistance than the transistor that is turning ON during the transition. Moreover, the logical effort of NAND gates is even lower because velocity saturation has a smaller effect on series nMOS transistors that see only part of the electric field between drain and source as compared to a single nMOS transistor that experiences the entire field. This effect is less significant for NOR gates because pMOS transistors have lower mobility and thus experience less velocity saturation. The efforts are fairly consistent across process and voltage. In comparison, the velocity-saturated model from Example 4.12 predicts logical efforts of 1.20, 1.39, 1.50, and 2.00 for NAND2, NAND3, NOR2, and NOR3 gates, agreeing reasonably well with the nanometer processes. The parasitic delays show greater spread because of the variation in the relative capacitances of diffusion and gates.

This data includes more detail than the designer typically wants when doing design by hand; the coarse estimates of logical effort from Table 4.2 are generally sufficient for an initial design. However, the accurate delay vs. fanout information, often augmented with input slope dependence, is essential when characterizing a standard cell library to use with

TABLE 8.8 Logical effort and parasitic delay of gates in various processes

Vendor		Orbit	HP	AMI	AMI	TSMC	TSMC	TSMC	IBM	IBM	IBM
Model		MOSIS	MOSIS	MOSIS	MOSIS	MOSIS	MOSIS	TSMC	IBM	IBM	IBM
Feature Size f	nm	2000	800	600	600	350	250	180	130	90	65
V_{DD}	V	5	5	5	3.3	3.3	2.5	1.8	1.2	1.0	1.0
FO4 Delay	ps	856	297	230	312	210	153	75.6	46.0	37.3	17.2
τ	ps	170	59	45	60	40	30	15	9.0	7.4	3.3
Logical Effort											
Inverter		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
NAND2		1.13	1.07	1.05	1.08	1.12	1.12	1.14	1.16	1.20	1.26
NAND3		1.32	1.21	1.19	1.24	1.29	1.29	1.31	1.35	1.41	1.51
NAND4		1.53	1.37	1.36	1.42	1.47	1.47	1.50	1.55	1.62	1.78
NOR2		1.57	1.59	1.58	1.60	1.52	1.50	1.50	1.57	1.56	1.50
NOR3		2.16	2.23	2.23	2.30	2.07	2.02	2.00	2.12	2.08	1.96
NOR4		2.76	2.92	2.96	3.09	2.62	2.52	2.53	2.70	2.60	2.43
Parasitic Delay											
Inverter		1.08	1.05	1.18	1.25	1.33	1.18	1.03	1.16	1.07	1.20
NAND2		1.87	1.85	1.92	2.10	2.28	2.07	1.90	2.29	2.25	2.47
NAND3		3.34	3.30	3.40	3.79	4.15	3.65	3.51	4.14	4.10	4.44
NAND4		4.98	5.12	5.22	5.78	6.30	5.47	5.52	6.39	6.39	6.79
NOR2		2.86	2.91	3.29	3.56	3.52	2.95	2.85	3.35	3.01	3.29
NOR3		5.65	6.05	7.02	7.70	6.89	5.61	5.57	6.59	5.76	6.35
NOR4		9.11	10.3	12.4	13.9	11.0	8.76	8.95	10.54	9.11	10.16

a static timing analyzer. The FO4 inverter delays may differ slightly from Table 8.5 because the widths of the transistors are different.

8.5.4 Power and Energy

Recall from Section 5.1 that energy and power are proportional to the supply current. They can be measured based on the current out of the power supply voltage source. For example, the following code uses the `INTEGRAL` command to measure charge and energy delivered to a circuit during the first 10 ns.

```
.measure charge INTEGRAL I(vdd) FROM=0ns TO=10ns
.measure energy param='charge*SUPPLY'
```

Alternatively, HSPICE allows you to directly measure the instantaneous and average power delivered by a voltage source.

```
.print P(vdd)
.measure pwr AVG P(vdd) FROM=0ns TO=10ns
```

Sometimes it is helpful to measure the power consumed by only one gate in a larger circuit. In that case, you can use a separate voltage source for that gate and measure power only from that source. Unfortunately, this means that `vdd` cannot be declared as `.global`.

When the input of a gate switches, it delivers power to the supply through the gate-to-source capacitances. Be careful to differentiate this input power from the power drawn by the gate discharging its internal and load capacitances.

8.5.5 Simulating Mismatches

Many circuits are sensitive to mismatches between nominally identical transistors. For example, sense amplifiers (see Section 12.2.3.3) should respond to a small differential voltage between the inputs. Mismatches between nominally identical transistors add an offset that can significantly increase the required voltage. Merely simulating in different design corners is inadequate because the transistors will still match each other. As discussed in Section 7.5.2, the mismatch between currents in two nominally identical transistors can be primarily attributed to shifts in the threshold voltage and channel length. Figure 8.31 shows an example of simulating this mismatch. Each transistor is replaced by an equivalent circuit with a different channel length and a voltage source modeling the difference in threshold voltage. Note that many binned BSIM models do not allow setting the transistor length shorter than the minimum value supported by the process. Obtaining data on parameter variations was formerly difficult but is now part of the vendor's model guide in nanometer processes.

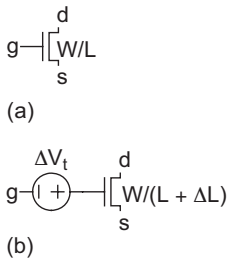

FIGURE 8.31
Modeling mismatch

In many cases, the transistors are not adjacent and may see substantial differences in voltage and temperature. For example, two clock buffers in different corners of the chip that see different environments will cause skew between the two clocks. The voltage difference can be modeled with two different voltage sources. The temperature difference is most easily handled through two separate simulations at different temperatures.

8.5.6 Monte Carlo Simulation

Monte Carlo simulation can be used to find the effects of random variations on a circuit. It consists of running a simulation repeatedly with different randomly chosen parameter offsets. To use Monte Carlo simulation, the statistical distributions of parameters must be part of the model. Manufacturers commonly supply such models for nanometer processes.

For example, consider modifying the FO4 inverter delay simulation from Figure 8.10 to obtain a statistical delay distribution. The transient command must be changed to

```
.tran 1ps 1000ps SWEEP MONTE=30
```

The `.measure` statements report average, minimum, maximum, and standard deviation computed from the 30 repeated simulations. The mean is 17.1 ps and the standard deviation is $\sigma = 0.56$ ps.

Good models will include parameters that the user can set to control whether die-to-die variations, within-die variations, or both are considered. They also may accept information extracted from the layout such as transistor orientation and well edge proximity.

8.6 Interconnect Simulation

Interconnect parasitics can dominate overall delay. When an actual layout is available, the wire geometry can be extracted directly. If only the schematic is available, the designer may need to estimate wire lengths. For small gates, even the capacitances of the wires inside the gate are important. Therefore, some companies use *parasitic estimator* tools to

guess wire parasitics in schematics based on the number and size of the transistors. In any case, the designer must explicitly model long wires based on their estimated lengths in the floorplan.

Once wire length and pitch are known or estimated, they can be converted to a wire resistance R and capacitance C using the methods discussed in Section 6.2. A short wire (where wire resistance is much less than gate resistance) can be modeled as a lumped capacitor. A longer wire can be modeled with a multisegment π-model. A four-segment model such as the one shown in Figure 8.32 is generally quite accurate. The model can be readily extended to include coupling between adjacent lines.

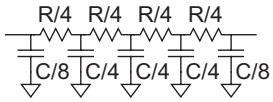

FIGURE 8.32 Four-segment π model for interconnect

In general, interconnect consists of multiple interacting signal and power/ground lines [Young00]. For example, Figure 8.33(a) shows a pair of parallel signals running between a pair of ground wires. Although it is possible to model the ground lines with a resistance and inductance per unit length, it is usually more practical to treat the supply networks as ideal, then account for power supply noise separately in the noise budget. Figure 8.33(b) shows an equivalent circuit using a single π-segment model. Each line has a series resistance and inductance, a capacitance to ground, and mutual capacitance and inductance. The mutual elements describe how a changing voltage or current in one conductor induce a current or voltage in the other.

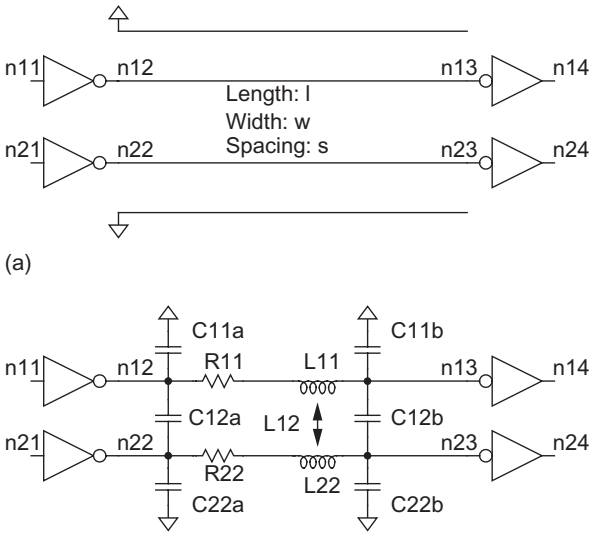

FIGURE 8.33 Lossy multiconductor transmission lines

HSPICE also supports the `W` element that models lossy multiconductor transmission lines. This is more convenient than constructing an enormous π-model with resistance, capacitance, inductance, mutual capacitance, and mutual inductance. Moreover, HSPICE has a built-in two-dimensional field solver that can compute all of the terms from a cross-sectional description of the interconnect. Figure 8.34 gives a SPICE deck that uses the field solver to extract the element values and models the lines with the `W` element.

The deck describes a two-dimensional cross-section of the interconnect that the field solver uses to extract the electrical parameters. The interconnect consists of the two signal traces between two ground wires. Each wire is 2 μm wide and 0.7 μm thick. The copper wires are sandwiched with 0.9 μm of low-k ($\varepsilon = 3.55\varepsilon_0$) dielectric above and below. The $N = 2$ signal traces are spaced 6 μm from the ground lines and 2 μm from each other and have a length of 6 mm. The HSPICE field solver is quite flexible and is fully documented in the HSPICE manual. It generates the transmission line model and writes it to the `coplanar.rlgc` file. The file contains resistance, capacitance, and inductance matrices and is shown in Figure 8.35.

The matrices require a bit of effort to interpret. They are symmetric around the diagonal so only the lower half is printed. The resistances are $R_{11} = R_{22} = 12.4\ \Omega$/mm. The inductances are $L_{11} = L_{22} = 0.67$ nH/mm and $L_{12} = 0.37$ nH/mm. The capacitance matrix represents coupling capacitances with negative numbers and places the sum of all the capacitances for a wire on the diagonal. Therefore, $C_{11} = C_{22} = 0.0117$ pF/mm and $C_{12} = 0.0137$ pF/mm. In the π-model, half of each of these capacitances is lumped at each end.

Figure 8.36 shows the voltages along the wires. The characteristic velocity of the line is approximately $1/\sqrt{L_{11}(C_{11}+C_{12})} = 2.4\times10^{11}$ mm/s. This is close to the speed of light (3×10^{11} mm/s) because the model assumes air rather than a ground plane outside the dielectric. The flight time down the wire is 6 mm/(2.4×10^{11} mm/s) = 25 ps.

```
* interconnect.sp
*-----------------------------------------------------------------------
* Parameters and models
*-----------------------------------------------------------------------
.param SUPPLY=1.0
.include '../models/ibm065/models.sp'
.temp 70
.option post
*-----------------------------------------------------------------------
* Subcircuits
*-----------------------------------------------------------------------
.global vdd gnd
.subckt inv a y N=100nm P=200nm
M1   y   a   gnd   gnd   NMOS   W='N'   L=50nm
+ AS='N*125nm' PS='2*N+250nm' AD='N*125nm' PD='2*N+250nm'
M2   y   a   vdd   vdd   PMOS   W='P'   L=50nm
+ AS='P*125nm' PS='2*P+250nm' AD='P*125nm' PD='2*P+250nm'
.ends
*-----------------------------------------------------------------------
* Compute transmission line parameters with field solver
*-----------------------------------------------------------------------
.material     oxide         DIELECTRIC          ER=3.55
.material     copper        METAL               CONDUCTIVITY=57.6meg
.layerstack   chipstack     LAYER=(oxide,2.5um)
.fsoptions    opt1          ACCURACY=MEDIUM     PRINTDATA=YES
.shape        widewire      RECTANGLE           WIDTH=2um    HEIGHT=0.7um
.model        coplanar      W                   MODELTYPE=FieldSolver
+ LAYERSTACK=chipstack      FSOPTIONS=opt1      RLGCFILE=coplanar.rlgc
+ CONDUCTOR=(SHAPE=widewire ORIGIN=(0,0.9um)    MATERIAL=copper TYPE=reference)
+ CONDUCTOR=(SHAPE=widewire ORIGIN=(8um,0.9um)  MATERIAL=copper)
+ CONDUCTOR=(SHAPE=widewire ORIGIN=(12um,0.9um) MATERIAL=copper)
+ CONDUCTOR=(SHAPE=widewire ORIGIN=(20um,0.9um) MATERIAL=copper TYPE=reference)
*-----------------------------------------------------------------------
* Simulation netlist
*-----------------------------------------------------------------------
Vdd  vdd   gnd   'SUPPLY'
Vin  n11   gnd   PULSE   0 'SUPPLY' 0ps 20ps 20ps 500ps 1000ps
W1   n12   n22   gnd    n13   n23   gnd   FSmodel=coplanar N=2 l=6mm
X1   n11   n12   inv    M=80
X2   n13   n14   inv    M=40
X3   gnd   n22   inv    M=80
X4   n23   n24   inv    M=40
*-----------------------------------------------------------------------
* Stimulus
*-----------------------------------------------------------------------
.tran 1ps 250ps
.end
```

FIGURE 8.34 SPICE deck for lossy multiconductor transmission line

```
* L(H/m), C(F/m), Ro(Ohm/m), Go(S/m), Rs(Ohm/(m*sqrt(Hz)), Gd(S/(m*Hz))
.MODEL coplanar W MODELTYPE=RLGC, N=2
+ Lo =  6.68161e-007
+       3.67226e-007  6.68161e-007
+ Co =  2.53841e-011
+       -1.36778e-011  2.53841e-011
+ Ro =  12400.8
+       0  12400.8
+ Go =  0
+       0  0
```

FIGURE 8.35 coplanar.rlgc file

When the input (n11) rises, the near end of the aggressor (n12) begins to fall. n12 levels out for a while at 0.2V as the driver supplies current to charge the rest of the wire. After one flight time (25 ps), the far end of the aggressor (n13) begins to fall. It undershoots to –0.2 V. After a second flight time, n12 levels out near 0. The far end oscillates for a while with a half-period of two flight times (50 ps).

When the aggressor falls, the victim is capacitively coupled down at both ends. The far end (n23) experiences stronger coupling because it is distant from its driver.

The *ringing* can be viewed as either the response of the 2nd order RLC circuit, or as a transmission line reflection. It is visible because the wires are far from their returns (hence having high inductance), are wide and thick enough to have low resistance (that would damp the oscillation), and are driven with an edge much faster than the wire flight time. If the inductance were reduced by moving the ground lines closer to the conductors, the ringing would decrease.

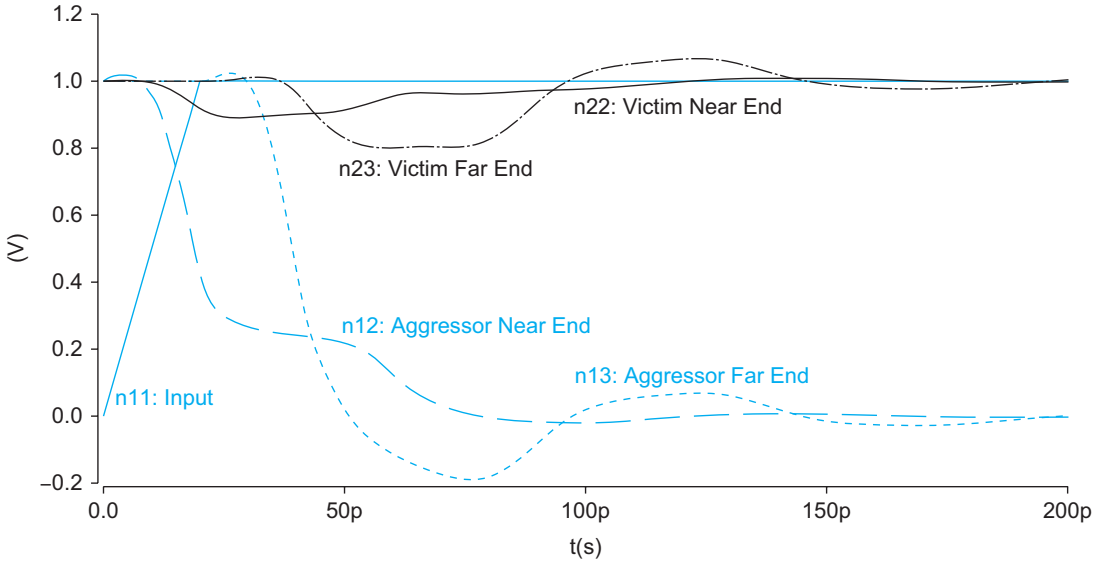

FIGURE 8.36 Transmission line response

8.7 Pitfalls and Fallacies

Failing to estimate diffusion and interconnect parasitics in simulations

The diffusion capacitance can account for 20% of the delay of an FO4 inverter and more than 50% of the delay of a high-fanin, low-fanout gate. Be certain when simulating circuits that the area and perimeter of the source and drain are included in the simulations, or automatically estimated by the models. Interconnect capacitance is also important, but difficult to estimate. For long wires, the capacitance and RC delay represent most of the path delay. A common error is to ignore wires while doing circuit design at the schematic level, and then discover after layout that the wire delay is important enough to demand major circuit changes and complete change of the layout.

Applying inappropriate input waveforms

Gate delay is strongly dependent on the rise/fall time of the input. For example, the propagation delay of an inverter is substantially shorter when a step input is applied than when an input with a realistic rise time is provided.

Applying inappropriate output loading

Gate delay is even more strongly dependent on the output loading. Some engineers, particularly those in the marketing department, report gate delay as the delay of an unloaded inverter. This is about one-fifth of the delay of an FO4 inverter or other gate with "typical" loading. When simulating a critical path, it is important to include the estimated load that the final stage must drive.

Choosing inappropriate transistor sizes

Gate delay also depends on transistor widths. Some papers compare a novel design with carefully selected transistor sizes to a conventional design with poorly selected sizes, and arrive at the misleading conclusion that the novel design is superior.

Identifying the incorrect critical path

During preliminary design, it is much more efficient to compare circuits by modeling only the critical paths rather than the entire circuit. However, this requires that the designer correctly identify the path that will be most critical; sometimes this requires much consideration.

Failing to account for hidden scale factors

Many CAD systems introduce scaling factors. For example, a circuit can be drawn with one set of design rules and automatically scaled to the next process generation. The CAD tools may introduce a scaling factor to reflect this change. Specifying the proper transistor sizes reflecting this scaling is notoriously tricky. Simulation results will look good, but mean nothing if scaling is not accounted for properly.

Blindly trusting results from SPICE

Novice SPICE users often trust the results of simulation far too much. This is exacerbated by the fact that SPICE prints results to many significant figures and generates pretty waveforms. As we have seen, there are a multitude of reasons why simulation results may not reflect the behavior of the real circuit.

When first using a new process or tool set, always predict what the results should be for some simple circuits (e.g., an FO4 inverter) and verify that the simulation matches expectation. It doesn't hurt to be a bit paranoid at first. After proving that the flow is correct, lock down all the models and netlist generation scripts with version control if possible. That way, if any changes are made, a good reason for the change must be evident and the simulations can be revalidated. In general, assume SPICE decks are buggy until proven otherwise. If the simulation does not agree with your expectations, look closely for errors or inadequate modeling in the deck.

Using SPICE in place of thinking

A related error, common among perhaps the majority of circuit designers, is to use SPICE too much and one's brain too little. Circuit simulation should be guided by analysis. In particular, designing to simulation results produced by the optimizer rather than designing based on understanding has led more than one engineer to grief.

Making common SPICE deck errors

Some of the common mistakes in SPICE decks include the following:

- Omitting the comment on the first line
- Omitting the new line at the end of the deck

- Omitting the `.option` post command when using a waveform viewer
- Leaving out diffusion parasitics
- Forgetting to set initial values for dynamic logic or sequential circuits

Using incorrect dimensions when `.option scale` is not set

If `.option scale` is not used, a transistor with W = 4, L = 2 would be interpreted as 4 by 2 meters! This often is outside the legal range of sizes in a BSIM model file, causing SPICE to produce error messages. Similarly, a drain diffusion of 3 × 0.5 μm should be specified as PD = 7u AD = 1.5p as opposed to the common mistakes of PD = 7 AD = 1.5 or PD = 7u AD = 1.5u.

Summary

When used properly, SPICE is a powerful tool to characterize the behavior of CMOS circuits. This chapter began with a brief tutorial showing how to perform DC and transient analyses to characterize and optimize simple circuits. SPICE supports many different transistor models. At the time of writing, the BSIM model is most widely used and describes MOSFET behavior quite well for most digital applications. When specifying the MOSFET connection, you must include not only the terminal connections (drain, gate, source, and body) and width and length, but also the area and perimeter of the source and drain that are used to compute parasitic capacitance.

Modern SPICE models have so many parameters that they are intractable for hand calculations. However, the designer can perform some simple simulations to characterize a process. For example, it is helpful to know the effective gate capacitance and resistance, the diffusion capacitance, and the threshold voltage and leakage current. You can also determine the delay of a fanout-of-4 inverter and the logical effort and parasitic delay of a library of gates to make quick estimates of circuit performance.

Most designers use SPICE to characterize real circuits. During preliminary design, you can model the critical path to quickly determine whether a circuit will meet performance requirements. A good model describes not only the circuit itself, but also the input edge rates, the output loading, and parasitics such as diffusion capacitance and interconnect. Most interconnect can be represented with a four-segment π model, although when inductance becomes important, the lossy multiconductor transmission line `W` element is convenient. Novel and "risky" circuits should be simulated in multiple design corners or with Monte Carlo analysis to ensure they will work correctly across variations in processing and environment. As SPICE is prone to garbage-in, garbage-out, it is often best to begin with a simple model and debug until it matches expectations. Then more detail can be added and tested incrementally.

Exercises

Note: This book's Web site at `www.cmosvlsi.com` contains SPICE models and characterization scripts used to generate the data in this chapter. Unless otherwise stated, try the exercises using the mosistsmc180 model file (extracted by MOSIS from test structures manufactured on the TSMC 180 nm process) in TTTT conditions.

8.1 Find the average propagation delay of a fanout-of-5 inverter by modifying the SPICE deck shown in Figure 8.10.

8.2 By what percentage does the delay of Exercise 8.1 change if the input is driven by a voltage step rather than a pair of shaping inverters?

8.3 By what percentage does the delay of Exercise 8.1 change if *X*5, the load on the load, is omitted?

8.4 Find the input and output logic levels and high and low noise margins for an inverter with a 3:1 *P/N* ratio.

8.5 What *P/N* ratio maximizes the smaller of the two noise margins for an inverter?

8.6 Generate a set of eight I-V curves like those of Figure 8.16–8.17 for nMOS and pMOS transistors in your process.

8.7 The `char.pl` Perl script runs a number of simulations to characterize a process. Use the script to add another column to Table 8.5 for your process.

8.8 The `charlib.pl` script runs a number of simulations to extract logical effort and parasitic delay of gates in a specified process. Add another column to Table 8.8 for your process.

8.9 Use the `charlib.pl` script to find the logical effort and parasitic delay of a 5-input NAND gate for the outermost input.

8.10 Exercise 4.10 compares two designs of 2-input AND gates. Simulate each design and compare the average delays. What values of x and y give least delay? How much faster is the delay than that achieved using values of x and y suggested from logical effort calculations? How does the best delay compare to estimates using logical effort? Let $C = 10\ \mu$m of gate capacitance.

8.11 Exercise 4.13 asks you to estimate the delay of a logic function. Simulate your design and compare your results to your estimate. Let one unit of capacitance be a minimum-sized transistor.